Robust Motion Planning for Mobile Robots

Paul Jacobs* LAAS-CNRS

John Canny† UC Berkeley

Abstract

In this work, we introduce a notion of robustness for a mobile robot trajectory, which captures the sense that there are "nearby" trajectories which are also collision-free and satisfy the kinematic constraints. We present an algorithm which is guaranteed to plan robust paths with running time $O(n^4 \log n + \frac{n^2}{\delta^2})$, where n is the number of obstacle vertices and δ is a measure of the robustness of the path. We also discuss the modifications required for the algorithm to allow for uncertainty in position.

1 Introduction

Quality measures used for evaluating trajectories have usually related to minimizing the distance traveled or the transit time. However, as planning algorithms are implemented for real-world systems, there must be an acknowledgment of the difficulty of control. Trajectories must also be evaluated on the basis of their robustness to tracking errors. When there are no constraints on the allowable motions, these errors can be accounted for by growing obstacles to include some safety margin. In the case of a mobile robot, this is not enough. A mobile robot will typically be subject to a nonholonomic kinematic constraint. This implies that the dimension of its configuration space, $\mathbf{R}^2 \times \mathbf{S}^1$, will be larger than the number of degrees of freedom, and therefore not all trajectories will be feasible. Thus, we must check that trajectories exist that allow recovery from errors in both position ($\mathbf{R}^2$) and *orientation* (S^1). It may occur that finding a collision-free recovery trajectory may entail starting the planning process from the beginning and will be too time-consuming to perform on-line. Therefore, allowing for simple recovery trajectories must be made a part of the planning algorithm by insuring that it finds a nominal trajectory which can tolerate errors in position and orientation. When we are searching for a path to be used in a real-world system it is more important that the path allows us to recover from errors than that it is the absolute shortest or the time optimal.

There are also theoretical reasons to require robustness of trajectories. While no lower bounds have been established for the problem of planning smooth paths in the plane, there are indications that it may be intractable to find an exact path. In this paper, we discuss approximation algorithms which address the problem. However, for an approximation algorithm to be well-posed, a neighborhood of feasible trajectories must exist around the trajectory being searched for.[1] It is this notion of a neighborhood around a path that we characterize as robustness.

In the algorithms given here and in [1], there has been no theoretical reason to allow for errors in position. This was not required to ensure the well-posedness of the approximation algorithm, because we already knew the form of the canonical trajectories and that they pass through points on the boundaries of the obstacles.[2] However, we may wish to allow for perturbations in position for practical reasons. In reality, errors in position and heading are not decoupled. Thus, it is useful to build this into a robust planning algorithm. We present a simple extension to the basic algorithm which allows for robustness in position as well as orientation.

We analyze the configuration space obstacles for trajectories which satisfy the non-holonomic constraints imposed by the kinematics of a mobile robot. We use this characterization as the basis for a plane sweep algorithm which computes the configuration space obstacle for a trajectory segment in time $O(n^2 \log n)$, where n is the number of obstacle vertices in the representation of the obstacles. We present an algorithm to generate robust paths with running time $O(n^4 \log n + \frac{n^2}{\delta^2})$, where δ is a measure of the robustness of the path.

1.1 Previous Work

The specific problem we consider in this paper was first addressed by Dubins [2], who gives the form of the shortest bounded curvature path in the absence of obstacles. After him, there was a hiatus of nearly thirty years, until Laumond published work on the problem in the case that obstacles in the workspace must be avoided [3]. Unfortunately, the method presented there is not guaranteed to find a path. In Fortune and Wilfong [4], a decision algorithm is given to decide if a path exists under given conditions. The algorithm is exact, but does not generate the path in question. This algorithm runs in time doubly exponential in the complexity of the environment. Their work also introduces the notion, used here, of forming a configuration space for trajectories. The algorithm we have presented in [1], is an approximation algorithm to solve the problem.

Laumond [5] has presented fundamental work which we can relate to the notion of recovery trajectories. He shows that if there exists a path which connects two points of the configuration space, then there exists a path which respects the nonholonomic kinematic and maximum curvature constraints imposed by the mobile robot. However, the trajectories which allow the robot to move from one configuration to another will, in general, consist of many back and forth maneuvers. Furthermore, the number of these maneuvers required is unknown. In fine motion planning,

*Research supported by the David and Lucile Packard Foundation and by Groupement Robots d'Intervention sur Site Planetaire (RISP). This author is at the Laboratoire d'Automatique et d'Analyse des Systems, 7, Avenue du Colonel Roche, 31077 Toulouse Cedex, France.

†Research supported by the David and Lucile Packard Foundation and by PYI grant number IRL-8958577. This author is with the Dept. of Computer Science at the University of California at Berkeley.

[1]Since it is by definition that, in general, the approximate trajectory can not exactly track the optimal.

[2]Thus a trajectory approximating an optimal trajectory will also pass through the boundaries of the same obstacles.

the notion of error detection and recovery is due to Lozano-Perez, Mason and Taylor [6]. The concepts embodied in this framework are somewhat different than those we consider here.

2 Review of Terms and Theorems

In [1], we presented an algorithm for planning paths for a mobile robot subject to the nonholonomic kinematic constraint imposed by the Pfaffian $dx \tan\theta - dy = 0$ and the constraint imposed by a bound on the change in θ. In this section, we review the major results and terminology from that paper which will be needed here.

2.1 Statement of the Problem

Here we give a precise formulation of the general problem of planning minimal-length (equivalently, minimal-time) unit speed paths with bounded average curvature.

Problem Statement

Let $\Omega \subset \mathbf{R}^2$ be a (closed) set of polygons in the plane with a finite number of vertices. This set represents the obstacles in the environment. $\partial\Omega$ represents the boundary of the set of obstacles and $\mathrm{int}(\Omega)$ denotes its interior. The variable R is the minimum turning radius of the mobile robot.[3] We consider the class of paths $X(t) \in \mathbf{R}^2$ which satisfy the following constraints.[4]

1. $X(0) = \mathrm{IP}$, $\theta(0) = \theta_0$.[5]
2. $\exists t_f > 0$, such that $X(t_f) = FP$, $\theta(t_f) = \theta_f$.
3. $X(t) \in \mathbf{R}^2 \setminus int(\Omega)$, $\forall t \in [0, t_f]$.
4. $\|\dot{X}(t)\| = (\dot{x}^2(t) + \dot{y}^2(t))^{\frac{1}{2}} = 1$, $\forall t \in [0, t_f]$.
5. Given $R > 0$, $\|\dot{X}(t_1) - \dot{X}(t_2)\| \le R^{-1}|t_1 - t_2|$, $\forall t_1, t_2 \in [0, t_f]$.

The problem is to find a path $X^*(t)$ defined for $t \in [0, t_f^*]$ such that t_f^* is minimal over all paths in this class.

2.2 Relevant Theorems from Previous Work

In this section, we present some theorems which relate to the method of motion planning for mobile robots given in [1]. We will refer back to these later in this paper.

The following corollary to Proposition 1 of Dubins[2] allows us to consider only shortest paths in our planning method.

Corollary 2.1 ([7]) *If $\exists X$ satisfying the conditions of the problem statement with finite length, then $\exists X^*$ of minimal length which also satisfies the conditions.*

Definition 2.2 *A* **simple path** *is a a C^1 curve which is either*

1. *an arc of a circle of radius R, followed by a line segment, followed by an arc of a circle of radius R.*
2. *a sequence of three arcs of circles of radius R.*
3. *a subpath of a path of either of these two types.*

The following corollary to Theorem 1 of Dubins[2] gives us the form of the shortest paths.

Corollary 2.3 ([7]) *Every minimal length planar curve X^* for which*

1. *The constraints of the problem statement are satisfied.*
2. *$X^*(0), X^*(t_f^*) \in (\mathbf{R}^2 \setminus \Omega) \cup \partial\Omega$.*
3. *$X^*(t) \in (\mathbf{R}^2 \setminus \Omega)$, $\forall t \in]0, t_f^*[$.*

[3]The maximum average curvature of the path is therefore R^{-1}.

[4]This general formulation of the problem is due to Dubins [2].

[5]If we describe the path $X(t) \in \mathbf{R}^2$ in coordinates by $(x(t), y(t))$, we can compactly represent the velocity vector in this system by $\theta(t) = \mathrm{atan2}(\dot{y}(t), \dot{x}(t))$, where the function $\mathrm{atan2}(\alpha, \beta)$ gives the arctangent respecting the signs of α and β.

is, $\forall t \in [0, t_f^*]$, necessarily a simple path.

Proof. The proof of this corollary consists of three parts. We sketch these parts here. It follows the basic outline of the proof given by Dubins for the case when there are no obstacles in [2]. We assume, throughout, without loss of generality that the minimum radius of curvature of a path is 1.

Step 1 Suppose there exists a path $X^*(t)$ of minimal length t_f^* satisfying the constraints of the problem statement such that $X^*(t) \in U$, $\forall t \in [0, t_f^*]$, where U is open and disjoint from Ω. In Steps 1 and 2, we show that $X^*(t)$ must be a simple path. Because the set $\{X^*(t) | t \in [0, t_f^*]\}$ is compact there is a minimal distance ϵ between it and the boundary of U. Because U is open $\epsilon > 0$. Now partition the curve $X^*(\cdot)$ into pieces of length $\frac{\epsilon}{2}$. Between the endpoints of each piece, the shortest path must be the minimal length path without obstacles (because the length of each path is less than ϵ it must lie entirely within U.) By Dubins [2], this is a simple path.

Step 2 We have shown that the path $X^*(t)$ of Step 1 is composed of arcs of unit circles and line segments. Following the proof of Dubins [2], in order to show that $X^*(t)$ is, in fact, a simple path itself reduces to showing that any sequence of four arcs (denoted CCCC), two arcs and a line (CCL), or a line followed by an arc followed by a line (LCL) can not be on a minimal length path. We do so by showing that, in each case, there exists a homotopy (a continuous deformation) which respects the curvature constraints and shortens the path. Because the points of the deformed curve can be arbitrarily close to the original, the deformed curve must also lie in U. Thus the original could not be of minimal length.

Step 3 Now suppose there exists a minimal length path $X^*(t)$ satisfying the theorem statement. In particular, $X^*(t)$ is disjoint from Ω except possibly at $X^*(0)$ or $X^*(t_f^*)$. Given any $\delta > 0$ the path $X^*(t)$ for $t \in [\delta, t_f^* - \delta]$ lies in an open set which is disjoint from Ω, and thus must be a simple path. By continuity, $X^*(t)$ must be a simple path. ■

We present here the deformation of the CCL type paths. The others can be found in [2, 7].

Lemma 2.4 *The arc-length of any curve of type CCL can be reduced by a homotopy which respects the curvature constraints.*

Proof. We start by considering the case that the second circular arc has length greater than π. Without loss of generality, we place the first circle with origin $(0, 0)$. We consider the line $y = l$. We place the initial point of the curve at $(-1, 0)$ and the the final point at $(-4, l)$.[6] We assume that the initial direction of travel is in the clockwise sense.

We consider the path given by the placement of a third circle with center $(x, 1 + l)$, and then placing the second circle to match, such that the path along the second circle is of length at least π. When $x = \sqrt{-(l+1)(l-3)}$, the second circle is tangent to the line and the path does not traverse the third circle at all. This is the configuration which yields the CCL path. The deformation is illustrated in Figure 1. Three different values of x are shown, the one on the farthest right corresponds to the CCL path.

We will only analyze the cases for which $3 > l \ge -1$ since the problem is symmetric. We will show that the length of the path $D(x, l)$ monotonically decreases for x less than this value.[7]

$$D(x, l) : \;=\; \frac{3\pi}{2} + x + 4 + 4\arccos(d/4) \qquad (2.5)$$

[6]Any change in these has the effect of adding a constant to the length of the path.

[7]The distance function given only holds for x between $-\sqrt{-(l+5)(l-3)}$ and $\sqrt{-(l+1)(l-3)}$. For x larger than the upper point, there is a 2π jump in the path length. For x lower than the lower bound, there is no possible second circle.

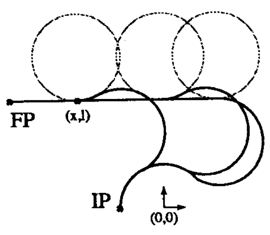

Figure 1: Deformation of the CCL Path

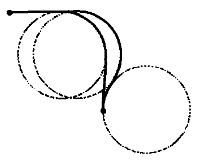

Figure 2: Deformation of Short CCL Paths

$$d = \sqrt{x^2 + (1+l)^2} \tag{2.6}$$

where d is the the distance between the centers of the first and third circles. By taking the first and second derivatives of $D(x,l)$, we find that the critical point of this function is at $\sqrt{-(l+1)(l-3)}$ and it is a maxima. Thus, the CCL path is always longer than the nearby deformed path obtained by reducing x.

Now we consider the simpler case that the second circle has an arc length less than or equal to π. In this case we can move the second circle along the line to shorten the straight line portion, and then connect the first and second circles by means of the straight line tangent to both which respects the orientation. This clearly reduces the arc length. Thus, we have a deformation which shortens the path for this case. The deformation is illustrated in Figure 2. ■

3 A Notion of Robustness

We make use of Corollaries 2.1 and 2.3 in order to characterize trajectories for a mobile robot. Together, these state that if a trajectory exists which satisfies the curvature constraints, then one exists which consists of sequences of simple paths alternating with (possibly single point) portions of the obstacle boundaries. We consider such paths as a canonical set of mobile robot trajectories. That is, finding a path reduces to searching within this set (see [1] for such a search algorithm). In this work, we introduce a notion of robustness which is associated with these canonical trajectories.

3.1 Robustness in Position

The notion of planning trajectories which allow for inaccuracies in the model of the environment or in locating the robot in its workspace is not a new one. Typically, these errors are accounted for by allowing a liberal growth of the obstacles by convolving them with a disk which represents the size of the uncertainty. We do the same in this work. However, due to the constraints on the turning radius, the application of this technique is not entirely straightforward when we perform the search for a trajectory in the set of canonical paths. We note that the obstacles no longer have the form of polygons.[8] The boundary of the grown obstacle consists of circular arcs of radius R_r around the vertices connected by line segments parallel to the original edges. In Section 4.1, we discuss the additions which must be made to the search algorithm of [1] to account for a possible position error of size R_r.[9]

3.2 Robustness in Orientation

It is in the notion of robustness to orientation errors that we make use of the canonical trajectories. First of all, the grown obstacle boundaries are smooth. This implies that, at the point a simple path leaves an obstacle boundary, the orientation of the robot is fixed. Positions on the boundary of the grown obstacle encode not only the location of the robot but also its direction, because, to have a smooth path, the disk of radius R_r must pass either tangent to an edge of the original obstacle or, if it contacts at a vertex, it must locally travel along the tangent to the circle at the point of contact.

The fact that each orientation is associated with a position on the obstacle boundary has the effect that an orientation error induces a change in position at the end of a simple path. Even so, we discuss robustness of the simple paths based only upon an interval of orientation. That is, we denote the position along a grown vertex by the angle to that point from the original polygon vertex.[10]

Definition 3.7 *A simple path is* **robust in orientation** *provided that*

1. *it is collision-free*
2. *if a small perturbation in orientation (position) is made at each point where the path passes through a grown vertex (edge), the path will remain collision-free, and will continue to pass in order through the same set of obstacle vertices and edges.*

4 Generating Robust Paths

In this section, we present an algorithm which is guaranteed to generate paths which are robust to errors in both position and orientation. In the grid-based algorithm of [1], we never explicitly computed the regions generated by mapping the workspace obstacles into each configuration space. The grid algorithm only required that we find slices of the configuration space obstacle. In order to guarantee robustness in orientation, we determine the entire obstacle. We note that the types of constraints given in [1] remain the same. There are, however, changes in calculating them due to the growth of obstacles from the position robustness.

4.1 Planning with Grown Obstacles

Once the obstacle has been grown, motion planning can once again be done for a point, and Corollary 2.3 stating the form of the canonical trajectories applies. In order to search for the path in the set of canonical trajectories, we consider the simple paths passing between pairs of edges or grown vertices of the polygonal obstacles. As in [1], this search is performed by transforming the obstacles into a configuration space *for the simple paths*. But to have a configuration space, we must find the appropriate parametrization of the simple paths. Because the obstacles have

[8]Our proof of Corollary 2.3 is sufficiently general to allow for such a set Ω of obstacles.

[9]The same constructions clearly apply to the problem of planning for a robot whose shape is a disk of radius R_r. We will frequently use this analogy in the following discussions, because it is more easily visualized than a trajectory which is not allowed to violate some region.

[10]Because we must also allow for paths which touch along an obstacle edge, we include under the heading of robustness in orientation, errors in position where a simple path touches along an obstacle edge.

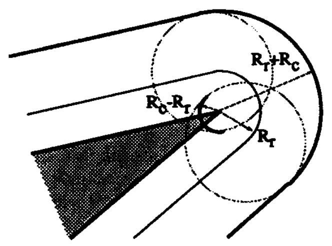

Figure 3: Possible Centers of Curvature when $R_c > R_r$

been grown, we must determine where the centers of curvature for a trajectory segment may lie. Once the centers of curvature and the type of simple path have been defined, the actual trajectory is a function of the location of the center of curvature at each endpoint.

For a simple path which leaves from a given point along a grown obstacle vertex, there are two possible initial directions of travel. Associated with each direction are two possible trajectory circles of radius R_c and associated loci of curvature centers which are circles of radius R_c+R_r and $|R_c-R_r|$ centered at the original vertex. The two circles correspond to left and right handed turns from the point on a grown vertex. There are two cases of interest, based on relative sizes of the minimum turning radius, R_c and the allowable position error, R_r.

When the turning radius is less than the radius of uncertainty, then only the centers of curvature on the circle with radius R_c+R_r are permitted, since any trajectory circle with a center on the $R_r - R_c$ circle will intersect the grown vertex in a neighborhood of the boundary point. When $R_c > R_r$, all four trajectory circles can contribute a portion of the simple path. Thus, both circles of curvature centers are allowed.

When the turning radius R_c is less than the size of the position uncertainty R_r, the minimal length trajectories may include portions of the grown obstacle vertices, which are arcs of circles of radius R_r. This is handled as a special case, by allowing the robot to pass from any point on the boundary of the obstacle to any other which lies in the correct direction.

Figure 3 depicts the possible locations of the center of curvature of a trajectory which passes through the boundary of the grown obstacle when the minimum radius of curvature,R_c, is larger than the dimension of the position error, R_r. The dark lines represent the boundary of the grown obstacle and the shaded lines represent the possible locations of the centers of curvature.

4.2 Generating Robustness in Orientation

In this section, we give an overview of the structure of the algorithm to generate paths which are robust to orientation errors. The operation of the algorithm is illustrated in Figure 4, where we search for a path passing from IP through A to FP. As shown, we represent the free configuration space by quadtrees. The lower configuration space represents all of the simple paths which start out turning left and end turning right and that pass between IP and A, with angles at IP on the horizontal axis and angles at A on the vertical. The upper configuration represents those paths from A to FP. We can see that by starting at θ_i we can reach the quadtree cell at the upper right hand corner of the lower configuration space. This translates into a range of orientations at point A, all of which can be reached following a collision-free simple path from (IP, θ_i). This range at A overlaps the larger quadtree blocks in the configuration space for A to FP. Thus all of the corresponding orientations at FP can be reached starting at any of the orientations at A, and hence from (IP, θ_i). If we construct a graph

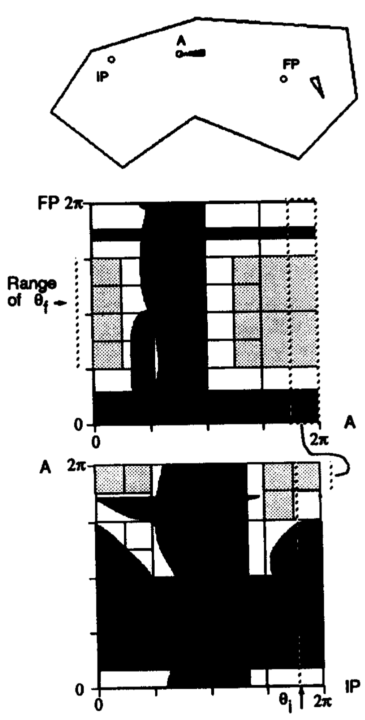

Figure 4: Quadtree in Configuration Space

whose nodes represent ranges of orientations at the points on the obstacle boundaries, and whose links represent the fact that the cartesian product of the two ranges overlap a free quadtree block in the appropriate configuration space, we can find a collision-free robust trajectory using a graph search.

The quadtree is a natural representation of regions in configuration space, because each quadtree cell represents the cartesian product of a range of orientations at each endpoint of the simple path. Therefore, the quadtree cells contained in a free region of configuration space have the nice property that, if a range of orientations ***overlaps*** a quadtree cell, all of the final orientations represented by that quadtree cell can be reached from that range. Therefore, in computing a quadtree representation of a configuration space, all subdivisions are made only in that space and do not affect the divisions required in any other space.[11]

4.3 Plane Sweep in Configuration Space

In order to compute a quadtree representation of configuration space, we must know how to transform the set of obstacles Ω into configuration space. The constraint curves discussed in [1] cut out regions in the configuration space. For all of the trajectories represented by the interior of such a region, the number of intersections with obstacles is constant. The parts of the constraint curves which form the obstacle boundary are all those pieces which separate a region with zero intersections from one with non-zero intersections. We can determine these pieces using a plane sweep algorithm.

In order to perform a plane sweep, we must know when the

[11]This property makes the algorithm trivially parallelizable.

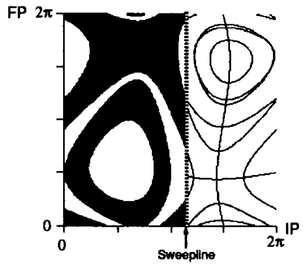

Figure 5: Plane Sweep in Configuration Space

sweepline intersects each constraint curve. As the swe line progresses, it will overlap different regions of the configuration space as illustrated in Figure 5. Although we envision the sweepline moving continuously across the configuration space, in reality it moves by jumping between consecutive angles at which the arrangement of the curves along the sweepline can change. To determine these **critical angles**, we must have a description of the form of the constraint curves. A tree holds the constraint curves which are currently crossed by the sweep line. The intervals of the sweep line between intersections with the curves correspond to connected regions of C-space cut out by the constraints. The regions may change their size and shape, but they appear and disappear only at the critical angles of the constraint curves which bound them. At each critical angle the sweep line structure is updated to reflect the changes in the affected regions.

We must also have a method to determine the number of intersections for a region as it is found during the plane sweep. This can be done because each critical angle is associated with a small number of constraint curves, and hence with a small number of edges in the environment. The basic idea is to keep a tally for each region of the number of intersections its trajectories make with the obstacles. When this number is non-zero, the region is part of the configuration space obstacle. The number of intersections for a new region can be determined in constant time from an adjacent region for which the number is known. This is because each constraint curve is associated with only a few edges in the environment. The techniques have been discussed in detail in [9].

4.4 Topology and Geometry of Constraints

The type of critical angle which is crossed determines the number and type of regions which result. In order to perform the sweep in time proportional to the number of critical angles, there must be a method for determining in constant time which critical angles cause qualitative changes in the boundary of the C-space obstacle. There are three types of critical angles at which the sweepline will enter or leave a region. They are 1) extrema of the constraint curves with respect to θ_i or θ_f, 2) endpoints of the curves, or 3) intersections between constraint curves. In this section, we show that it is possible to rigorously analyze the form of the curves in configuration space which are generated by the obstacles to determine the extrema. Similar procedures can be followed to determine the endpoints.

Once the obstacles are grown, the configuration spaces are of a different nature. They may have disconnected pieces, depending upon whether or not the obstacle was convex. However, the constraint curves are generated in exactly the same fashion as in [1]. That is, there are exactly four types of constraint curves. Some slight modifications must be made to allow for the circular arcs which have replaced the vertices. However, we note that when the constraints are computed using the dual formulation considering the location of the centers of curvature, then there is almost no change.

In this section, we will only examine the constraint in configuration space generated by the pairings of angles for which the arc-line-arc simple path is actually just an arc-arc simple path, that is the two circles are just touching. This constraint, called **Type D** in [1], forms the boundary of the region in configuration space in which certain simple paths may exist.

Suppose we have an arc-line-arc simple path which passes between two obstacle vertices. Let us consider a circle of radius R_c rolling along each grown vertex of an obstacle represented by an arc of radius R_r. Without loss of generality, we can place the first vertex at (0,0) and the second at (Δx,0). We write the center of the initial circle as $((R_c + R_r)\cos\theta_i, (R_c + R_r)\sin\theta_i)$ and the final as $(\Delta x + (R_c + R_r)\cos\theta_f, (R_c + R_r)\sin\theta_f)$. We consider the function $f(\theta_i, \theta_f)$ giving the square of the Euclidean distance between them. We know that the two circles are just touching when $f(\theta_i, \theta_f) = 4R_c^2$, thus the set $f^{-1}(4R_c^2)$ is the Type D curve.

We begin analyzing the form of the constraint curves using the techniques of differential topology. We will only use some simple results about the level sets of smooth functions. For the details, see [9]. In order to find the critical points of the curve we take the partial derivatives and set them equal to zero, while requiring $f(\theta_i, \theta_f) = 4R_c^2$.

$$\begin{aligned}\frac{\partial f}{\partial \theta_i} &= 2(R_c + R_r)\Delta x \sin\theta_i + \\ &\quad 2(R_c + R_r)^2 sin(\theta_i - \theta_f) = 0 \qquad (4.8)\\ \frac{\partial f}{\partial \theta_f} &= -2(R_c + R_r)\Delta x \sin\theta_f - \\ &\quad 2(R_c + R_r)^2 sin(\theta_i - \theta_f) = 0 \qquad (4.9)\end{aligned}$$

In order to find the critical angle of the first or last intersection of the sweepline with the Type D curve, we must know the location of the extrema with respect to θ_i and θ_f. In general, these occur in the case that one or the other of Equations 4.8 and 4.9 is satisfied. We can show that there exist angles (θ_i, θ_f) for which both partials are zero if the distance between the two vertices is exactly $2R_c$, $2R_r$, $4R_c + 2R_r$. In these cases, the Preimage theorem [10, p.21], states that $f^{-1}(4R_c^2)$ is not a smooth curve. These must be treated as special cases, but they can be resolved simply.

At this point, it appears that we must solve transcendental equations in order to find the extrema of the curves with respect to θ_i or θ_f. However, by recognizing the geometrical equivalents to Equations 4.8 and 4.9, we reduce the problem to one of finding the intersections between circles. By looking at projections implied by these equations we can show that the critical angles can be found by intersecting circles of radius $R_r + 3R_c$ and $R_r - R_c$ around one vertex with a circle of radius $R_c + R_r$ around the other. This gives a simple geometric characterization of these extremal points in terms of a quadratic equation describing the intersection of two circles.

Figure 6 depicts the Type D curves in configuration spaces for which the form of the curves fundamentally changes. These correspond to different separations between endpoints of the simple paths (which are vertices in this picture). For clarity, we have chosen, $R_c = 1$ and $R_r = 0$.

4.5 Complexity of the Search Algorithm

By the enumeration of critical angles, we can see that there are $O(n^2)$ of them. Because we must sort them, the plane sweep algorithm takes $O(n^2 \log n)$, where n is the number of corners in the environment, for each such piece of configuration space. All told generation of the $O(n^2)$ pieces of configuration space then takes time $O(n^4 \log n)$.

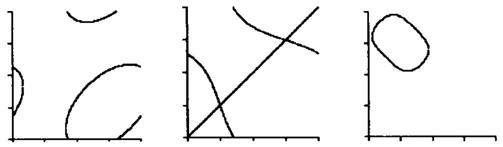

Figure 6: Form of the Type D constraint curves for different values of Δx. From left to right: $0 < |\Delta x| < 2$; $|\Delta x| = 2$; $2 < |\Delta x| < 4$. (Illustrated for $R_c = 1$ and $R_r = 0$.)

If we perform our search using Dijkstra's algorithm, and there are at most $O(\frac{n^2}{\delta^2})$ quadtree blocks, this is the complexity of the search, disregarding the cost of generating the quadtree blocks. This can be done during the plane sweep at a cost proportional to the number of blocks. Therefore, the search requires time $O(n^4 \log n + \frac{n^2}{\delta^2})$.

4.6 Robustness of the Paths

Using the quadtree-based algorithm, the search will yield a path which is robust to orientation perturbations. Furthermore, there are some nice properties to robust paths, besides the practical benefit of having a path which is less sensitive to errors in the trajectory following capability of actual mobile robot.

We first examine when the algorithm will find a robust path. In this discussion, we will consider spacing along a grown vertex. The case for edges is similar although the minimum spacing may differ. Suppose we can fit a square with sides length δ in free space. Such a square is associated with a robust simple path. The perturbation in orientation which can be tolerated is $\pm\frac{\delta}{2}$ on either side of the nominal angle. A complete path from IP to FP which consists of simple paths all tolerating at least this large of a perturbation is called a **δ-robust path**.

Because the quadtree-based algorithm makes use of a regular tesselation of the configuration spaces, we must consider the effect of choosing free blocks in the configuration space which are aligned with the quadtree edges. It turns out that it is not true that if a δ-robust path exists, that the algorithm will find it. This is because the algorithm will only search the set of δ-robust paths where the squares in configuration space are aligned with the quadtree blocks. However, if a δ-robust path exists, there exists an aligned $\frac{\delta}{2}$-robust path. This is because, in each configuration space, the δ-robust path is associated with an interval of initial angles of size δ. Such an interval must necessarily completely overlap at least one interval of length $\frac{\delta}{2}$ aligned with the quadtree. Because of the overlap property discussed in Section 4.2, any final angles which can be reached from an orientation in the interval of length δ must also be reachable from an orientation in the interval of length $\frac{\delta}{2}$.

The smaller blocks completely capture the connectivity of the path, although they do lose some of the robustness. One advantage however is that the same overlap property eliminates the necessity for looping more than a few times. It implies that if a δ-robust path passes through a vertex more than once and overlaps the same quadtree block of size $\frac{\delta}{2}$ then all of the trajectory between the first and last passage through this vertex can be eliminated. By the overlap property, no advantage is derived from visiting the vertex and overlapping a range more than once. The connectivity of the path preceding and following the loop is retained.

5 Consequences of Robustness

In this paper, we have presented a notion of robustness which is closely aligned with the concept of canonical trajectories given by Corollary 2.3. In concluding, we would like to point out what we see to be the benefits of this linkage and of robustness in general.

As we have noted above, the concept of robustness puts an a priori limit on the number of loops which can exist in a trajectory.

Because the idea of robustness to orientation errors has been defined with respect to the endpoints of simple paths, this gives an idea of critical points along the trajectory at which the robots performance can be compared against the error bounds.

We have been considering only the representation of robust paths in the configuration space for simple paths. However, by looking at the envelope generated by each robust simple path passing between two obstacle vertices or edges, we can see that we have essentially divided the real environment up into channels. Furthermore, because orientation errors are specified for both endpoints of the simple path, the induced position errors are independent of the size of the environment. Thus uncertainty in position can essentially be decoupled from orientation errors. In contrast, if the allowable orientation error was specified for only one endpoint, the associated position error increases the longer the path is followed.

Finally, the nominal trajectories are specified in a robust manner. That is, if a given nominal simple path ends in a right hand turn, then its robustness implies that, as long as the robot is remained within its error bounds, no sensing is needed along the trajectory to determine whether a right or left turn is required in order to fall within the robust interval; a right hand turn will always suffice. This is useful because such a decision would be very sensitive to errors in sensing.

References

[1] P. Jacobs and J. Canny, "Planning smooth paths for mobile robots," in *Proceedings of the 1989 International Conference on Robotics and Automation*, pp. 2–7, IEEE, May 1989.

[2] L. E. Dubins, "On curves of minimal length with a constraint on average curvature and with prescribed initial and terminal positions and tangents," *American Journal of Mathematics*, vol. 79, pp. 497–516, 1957.

[3] J.-P. Laumond, "Finding collision-free smooth trajectories for a non-holonomic mobile robot," in *IJCAI 87 Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, (Milan), pp. 1120–1123, IJCAI, Inc., August 1987.

[4] S. Fortune and G. Wilfong, "Planning constrained motion," in *STOCS*, (Chicago IL), pp. 445–459, Association for Computing Machinery, May 1988.

[5] J.-P. Laumond, "Feasible trajectories for mobile robots with kinematic and environment constraints," in *International Conference on Intelligent Autonomous Systems*, (Amsterdam), pp. 346–354, 1986.

[6] T. Lozano-Perez, M. Mason, and R. Taylor, "Automatic synthesis of fine motion strategies for robots," *International Journal of Robotics Research*, vol. 3, no. 1, 1984.

[7] P. Jacobs, "Minimal length curvature constrained paths in the presence of obstacles," LAAS/CNRS Report 90042, Laboratoire d'Automatique et d'Analyse des Systemes, February 1990.

[8] K. Mehlhorn, *Data Structures and Algorithms 3: Multidimensional Searching and Computational Geometry*. Berlin: Springer-Verlag, 1984.

[9] P. Jacobs, "Planning robot motion with dynamic constraints," Tech. Rep. ESRC 89-21/RAMP 89-17, University of California, Berkeley, Engineering Systems Research Center, October 1989. PhD Thesis.

[10] V. Guillemin and A. Pollack, *Differential Topology*. Prentice-Hall, 1974.