Kamal Kant
Steven W. Zucker*
Computer Vision and Robotics Laboratory
Department of Electrical Engineering
McGill University
Montréal, Québec, Canada H3A2A7

Toward Efficient Trajectory Planning: The Path-Velocity Decomposition

Abstract

We present a novel approach to solving the trajectory planning problem (TPP) in time-varying environments. The essence of our approach lies in a heuristic but natural decomposition of TPP into two subproblems: (1) planning a path to avoid collision with static obstacles and (2) planning the velocity along the path to avoid collision with moving obstacles. We call the first subproblem the path planning problem (PPP) and the second the velocity planning problem (VPP). Thus, our decomposition is summarized by the equation TPP ⇒ PPP + VPP. The symbol ⇒ indicates that the decomposition holds under certain assumptions, e.g., when obstacles are moving independently of (*i.e., not tracking*) *the robot. Furthermore, we pose the VPP in* path-time *space, where time is explicitly represented as an extra dimension, and reduce it to a graph search in this space. In fact, VPP is transformed to a two-dimensional PPP in* path-time *space with some additional constraints. Algorithms are then presented to solve the VPP with different optimality criteria: minimum length in path-time* space, *and minimum time.*

1. Introduction

How does one get from "here" to "there"? This is an everyday version of the trajectory planning problem. We solve it so often and so intuitively that we are seldom aware of the complexity underlying it. Our purpose in this paper is to illustrate this complexity in a manner that motivates a particular decomposition of spatial and temporal aspects. This leads to a new algorithm for solving the trajectory planning problem in time-varying environments, an algorithm that we summarize with the decomposition TPP ⇒ PPP + VPP. The *trajectory planning problem* consists of a static *path planning problem* and a temporal *velocity planning problem.*

1.1. Intuitive Motivation: A Cocktail Party

What are the essential ingredients of a trajectory planning problem (TPP) and how are they used? Consider the following scenario:

> Suppose that you arrive early at a cocktail party and the hall is sparsely filled. You desire a drink and glance around the hall to locate the bar. Except for avoiding an occasional piece of furniture, you walk straight toward it. However, as another person approaches your path, you slow down to avoid a collision and, after she passes, you speed up again.

This simple example illustrates the key intuition behind our approach in the presence of moving obstacles: *both the path and the velocity along the path are crucial for collision avoidance.** Furthermore, it suggests an intuitive decomposition of the whole problem into two subproblems: (1) planning the path to avoid collisions with the stationary obstacles and (2) planning the velocity along the path to avoid collisions with the moving obstacles (Kant and Zucker 1984). It is precisely this decomposition — how good it is, when it is valid, and how solvable the resulting subproblems are — that we are going to study in this paper.

* Senior Fellow, Canadian Institute for Advanced Research. This research was supported by NSERC grant A4470.

The International Journal of Robotics Research,
Vol. 5, No. 3, Fall 1986,
© 1986 Massachusetts Institute of Technology.

* In this paper we are concentrating on those mathematical aspects of the trajectory planning problem that are relevant to robotics applications. Other aspects, such as the cognitive side of planning and motivations that can be attributed to the obstacles (e.g., people to avoid in a particular social situation) will not be considered.

1.2. The Decomposition: "Where" and "When"

Most previous research on trajectory planning in robotics has concentrated on the subproblem of finding paths among static obstacles (Udupa 1977; Lozano-Perez and Wesley 1979; Brooks 1982; Brooks and Lozano-Perez 1983). There is no notion of *time* involved in such formulations, so they do not generalize.* Note that among static obstacles, one need only answer the question, "where"? Once a path has been found, the robot's velocity along it does not matter. Such static problems are what we refer to as *path planning problems* (PPPs).

In the presence of moving obstacles, however, both "where" and "when" questions are involved. In this case, the velocity does matter; of two different velocity functions on the same path, one may lead to a collision while the other may not. That is why we call the second subproblem the *velocity planning problem* (VPP), or the problem of planning the velocity along a given path to avoid collision with moving obstacles. Our approach to the TPP exploits a natural decoupling of the two questions; the PPP answers the "where" question, and the VPP answers the "when" question: TPP $\Rightarrow$ PPP + VPP. We shall now focus on the advantages of this decomposition of TPPs in certain robotics applications.

1.3. Reduction in Complexity: Applications in Robotics

It is well known that planning paths even among static obstacles (the PPP) is a computationally difficult problem (Reif 1979; Schwartz and Sharir 1982*a*; Hopcroft, Joseph, and Whitesides 1984). The complexity of the general PPP is exponential in the number of degrees of freedom of the robot and it is polynomial in the number of polynomial equations describing the obstacles.

* Our concern is *time* as formulated in collision avoidance problems, i.e., avoiding moving obstacles. Researchers have considered coordinating movements of independent bodies where time may be implicit (Schwartz and Sharir 1982*b*). There are other areas in robotics, such as control, where time is an important aspect (Paul 1981; Brady et al. 1983).

However, the exponents and constants are large from a practical point of view. The implementations of these general algorithms do not yet exist. One implemented algorithm for the two-dimensional PPP may take tens of minutes to execute (Brooks and Lozano-Perez 1983).* The full problem of planning trajectories among moving obstacles is even harder. One reason for this increase in complexity is the additional time dimension in the TPP. While we shall deal with complexity issues later, for now it suffices to note that the above observation—that path and velocity planning are decomposable under certain conditions—provides one opening through this additional complexity barrier in the TPP. Most importantly, it is an opening that has applications in robotics. The most obvious example is that of a robot moving along a fixed path, either a track or a corridor. Another example involves a manipulator transferring payloads from one place to another in the presence of obstacles. Often these obstacles are moving, as on a conveyor belt. The obstacle may be another manipulator sharing a common workspace (Wallace 1984).

1.4. Overview

To summarize the introductory remarks, our general approach will be to:

1. plan the path of the robot to avoid collisions with static obstacles,
2. choose one of the paths to reach the goal position, and
3. plan the velocity to avoid collisions with obstacles moving across the path.

The advantage of this decomposition is that it provides a more efficient solution to the problem. There are both theoretical and practical limits, however, that restrict the application of this decomposition paradigm to the general TPP. There are cases when this paradigm will not yield a collision-free trajectory even though one may exist. For example our approach may fail if an obstacle is moving along the same path as the

* However, fast approximate algorithms for solving some PPPs exist (Brooks 1982).

robot. Such coincidences are unlikely, however, unless the obstacles are purposefully tracking the robot.

In this paper we formally define the TPP for a point robot. We motivate the decomposition of the TPP from a mathematical point of view and give precise formal definitions for both the PPP and the VPP; we briefly overview current approaches to the PPP and opt for the one representing free space; we present an algorithm to solve the VPP by reducing it to a graph search. Having presented this framework, we suggest some extensions to solve the TPP for a polyhedral robot (either two-dimensional or three-dimensional). In conclusion, we extend our approach to uncertain environments.

2. The Trajectory Planning Problem (TPP) — The Case of a Point Robot

Assume that the robot is a point object. Given its initial position, its final position, and the trajectories of the obstacles, the problem is to determine the trajectory of the robot as a function of time to avoid collisions with the obstacles. We may formally state the above as follows:

Let $T = [t_i, t_f]$ denote the time interval over which the trajectory of the robot is to be planned. The symbol t_i denotes the initial time and t_f denotes the final time. The symbol x_i indicates the position of the robot at time t_i. The symbol x_f indicates the position of the robot at time t_f. Let O denote the set of obstacles (rigid bodies) $O_1, O_2, \ldots, O_n$. Given that the obstacles and their trajectories are known, $O(t) = \{O_1(t) \cup O_2(t) \cup \cdots \cup O_n(t)\} \subset E^3$ is known $\forall t \in T$, where $O(t)$ denotes the space occupied by the obstacles at time t. We interpret the mapping $x(\cdot)$ from T to E^3, the 3-D Euclidean space, as the trajectory of the robot, so $x(t)$ denotes the position vector of the robot at time t. Note that $x(\cdot)$ has to be continuous to be physically realizable. From now on, we will use the word *mapping* to imply continuous mapping. The TPP is then:

Find a mapping $x: T \rightarrow E^3$ with $x(t_i) = x_i$, $x(t_f) = x_f$, such that $x(t) \cap O(t) = \emptyset$, $\forall t \in T$.

2.1. Separating Space and Time—Decomposing TPP into PPP and VPP

There are two ways to view a mapping. The first is as a correspondence between a point in the domain and a point in the range. The second takes the domain in its entirety and considers its image under the mapping. These two views provide a natural framework for our decomposition of the TPP. The mapping $x(\cdot)$ comprises

1. its image set $\pi \triangleq x(T) = \{x(t) : t \in T\} \subset E^3$, and
2. the correspondence $x_\pi : T \rightarrow \pi$.

As you recall, the TPP poses two questions—"where" and "when." The first subproblem deals with the where part of the question and the second deals with the when. The image π is the path of the robot as a space curve, and the correspondence $x_\pi(\cdot)$ determines the motion of the robot along the path π. The solution to the first subproblem determines a path and the solution to the second determines the motion along this path. Therefore, $x_\pi(\cdot)$ is the trajectory obtained by using the decomposition, and $x_\pi(t)$ will denote the position of the robot at time t on path π.

How does the constraint $x(t) \cap O(t) = \emptyset$ map onto our decomposition? How does it affect the image set π and the correspondence $x_\pi(\cdot)$? Let the set of obstacles $O(t) = \{O^s\} \cup \{O^m(t)\}$, where O^s denotes the set of static obstacles (therefore, independent of t) and O^m denotes the set of moving obstacles. Substituting for $O(t)$, the constraint becomes

$$x(t) \cap \{O^s \cup O^m(t)\} = \emptyset \quad \forall t \in T,$$

$$\Leftrightarrow \quad x(t) \cap O^s = \emptyset \quad \wedge \quad x(t) \cap O^m(t) = \emptyset \quad \forall t \in T.$$

The original constraint is also decomposed into a logical *AND* of two subconstraints. Consider the first subconstraint:

$$x(t) \cap O^s = \emptyset, \quad \forall t \in T,$$

$$\Leftrightarrow \quad \{x(t): \ t \in T\} \cap O^s = \emptyset.$$

Since O^s is independent of t,

$$\Leftrightarrow \quad \pi \cap O^s = \emptyset.$$

Thus the first subconstraint leads directly to the PPP.

Find an image set π such that $\pi \cap O^s = \emptyset$. (PPP)

Now, consider the second subconstraint $x(t) \cap O^m(t) = \emptyset$. We control two factors to satisfy this subconstraint: the image set π and the correspondence $x_\pi(\cdot)$. $T \rightarrow \pi$. Assume that we choose a π that satisfies the first subconstraint. The second subconstraint leads to the velocity planning problem.

Given the image set π, find a correspondence

$x_\pi(\cdot): T \rightarrow \pi$, such that $x_\pi(t) \cap O^m(t) = \emptyset$. (VPP)

To summarize, we have formalized the TPP $\Rightarrow$ PPP + VPP decomposition. This implies that the problem of planning the trajectory of a point robot among obstacles is decomposed into two subproblems: planning the path so that a collision with static obstacles is avoided and planning the velocity along this path so that a collision with moving obstacles is avoided.

3. Solving the PPP and the VPP

With the TPP $\Rightarrow$ PPP + VPP decomposition established, we next give an algorithm for solving the VPP. Since this algorithm presumes a path, we will first briefly review current approaches to the PPP.

3.1. The PPP—Determining a Path

The aim of path planning is to come up with an explicit representation of the collision-free paths that a robot may take. Geometrically, we may view it in two complementary ways: planning the path so that it avoids the obstacles or planning the path so that it lies in the free space, i.e., physical space minus the obstacles. We will give an algorithm from each approach.

3.1.1. Vgraph Algorithm

In a two-dimensional polygonal world, a necessary condition for a minimum length path is for it to be composed of straight line segments connecting a subset of the vertices of the polygonal obstacles (Wangdahl, Pollock, and Woodward 1974; Lozano-Perez and Wesley 1979). For an intuitive proof, see the MinPath Theorem in Appendix II. A two-dimensional PPP is solved as follows. Consider the graph VG with the set of nodes $N = \{I, F\} \cup V$, where I corresponds to the initial position and F to the final position. V is the set of vertices of the polygonal obstacles. Construct the edge set E of all the edges (n_i, n_j) such that the straight line connecting node n_i to node n_j does not intersect any of the obstacles. This graph $VG(N, E)$ is called the *Vgraph* and the optimal (shortest) collision-free path π from I to F is given by the minimum cost (graph) path from node I to node F. The cost of an edge is given by the Euclidean metric. The Vgraph Algorithm is illustrated in Fig. 1.

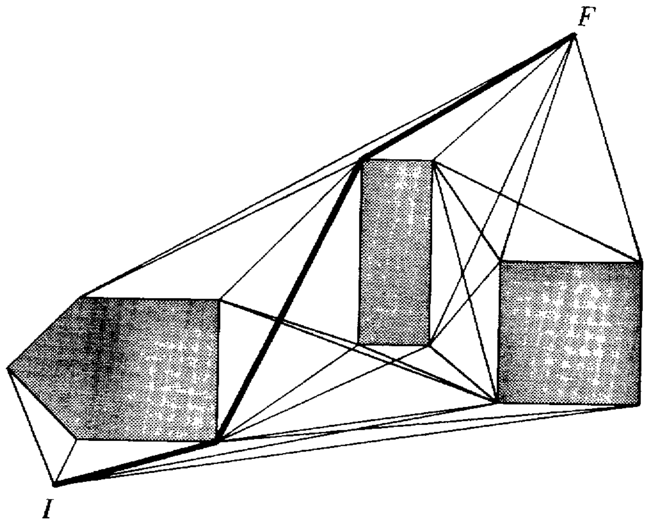


Fig. 1. The Vgraph algorithm to solve the two-dimensional PPP. The shaded polygons are obstacles. The shortest *collision-free path from* I *to* F *(shown in bold lines) is composed of straight line segments joining vertices of the obstacles.*

A version of this algorithm has been used by Moravac (1980) to plan paths for a robot among circular obstacles in a plane. The algorithm can be generalized to the 3-D case; however, it does not yield shortest paths. One approach is to introduce additional vertices along the edges of the obstacles (Lozano-Perez and Wesley 1979). Still, nonexponential algorithms are not currently known for finding shortest paths among 3-D polyhedral obstacles.*

* We thank one of the referees for pointing this out.

Fig. 2. The triangulation approach to solving the PPP. The bounding rectangle represents the total workspace. A path from I *to* F *is shown in bold lines. Contrast this with Fig. 1.*

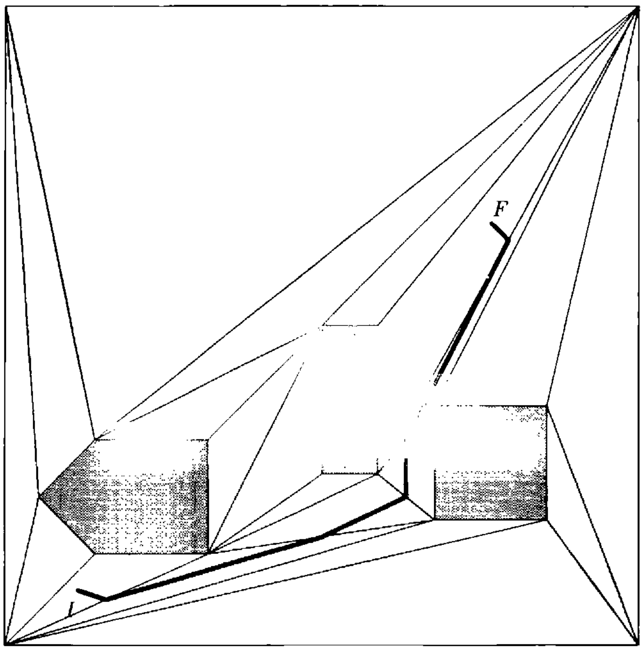
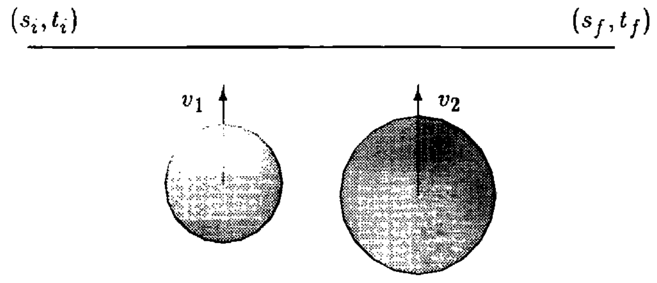


Fig. 3. A simple example of the VPP for a point robot. The given path is a straight line segment $\overline{s_i s_f}$. *The obstacles are the two circles moving with velocities* v_1 *and* v_2 *respectively. The robot is at position* s_i *at time* t_i. *Different collision-free velocity profiles may be determined to achieve different objectives. In one case, the objective may be to reach the final position* s_f *at a given time* t_f; *in another, the objective may be to reach the final position* s_f *in minimum time.*



3.1.2. Cellular Decomposition of Free Space

One problem with Vgraph-like algorithms is that the path passes too close to the obstacles. The second approach of explicitly representing the free space avoids this problem. The basic idea is to keep to the middle of free space. Such approaches have been used by Lozano-Perez (1980); Brooks (1982); and O'Dunlaing and Yap (1983).

Let $U \subset E^3$ be the set that represents the total workspace of the robot (the set of positions reachable by the robot but ignoring the obstacles). The free space is given by $F = U - O^s$. Then, the obstacle avoidance constraint, $\pi \cap O^s = \emptyset$, is equivalent to $\pi \in F$. The PPP is now equivalent to determining if the set F is connected.

Consider the case when the obstacles are polyhedra and the robot's workspace is a polyhedron. The free space is this polyhedron with polyhedral obstacle holes in it. This is illustrated in two-dimensions in Fig. 2.

How do we represent this free space F? In two-dimensions, Brooks (1982) used generalized cones and Donald (1983) used channels in free space. A variation (in 3-D) is to triangulate the free space in tetrahedral (triangular, two-dimensional) cells. The triangulation leads to a graph with nodes corresponding to the tetrahedra. Two nodes are joined by an edge if the corresponding tetrahedra are adjacent.* Since a tetrahedron has four faces, the graph will be a quadgraph. A solution to a particular instance of the PPP is obtained by an efficient search of this graph.

Note that a graph path corresponds to an equivalence class of physical paths. For the same graph path, there may be many physical paths. This is illustrated in two-dimensions in Fig. 2. Such a graph representation may be compared to the road network in a city. It describes the possible routes a robot may take to reach its destination.

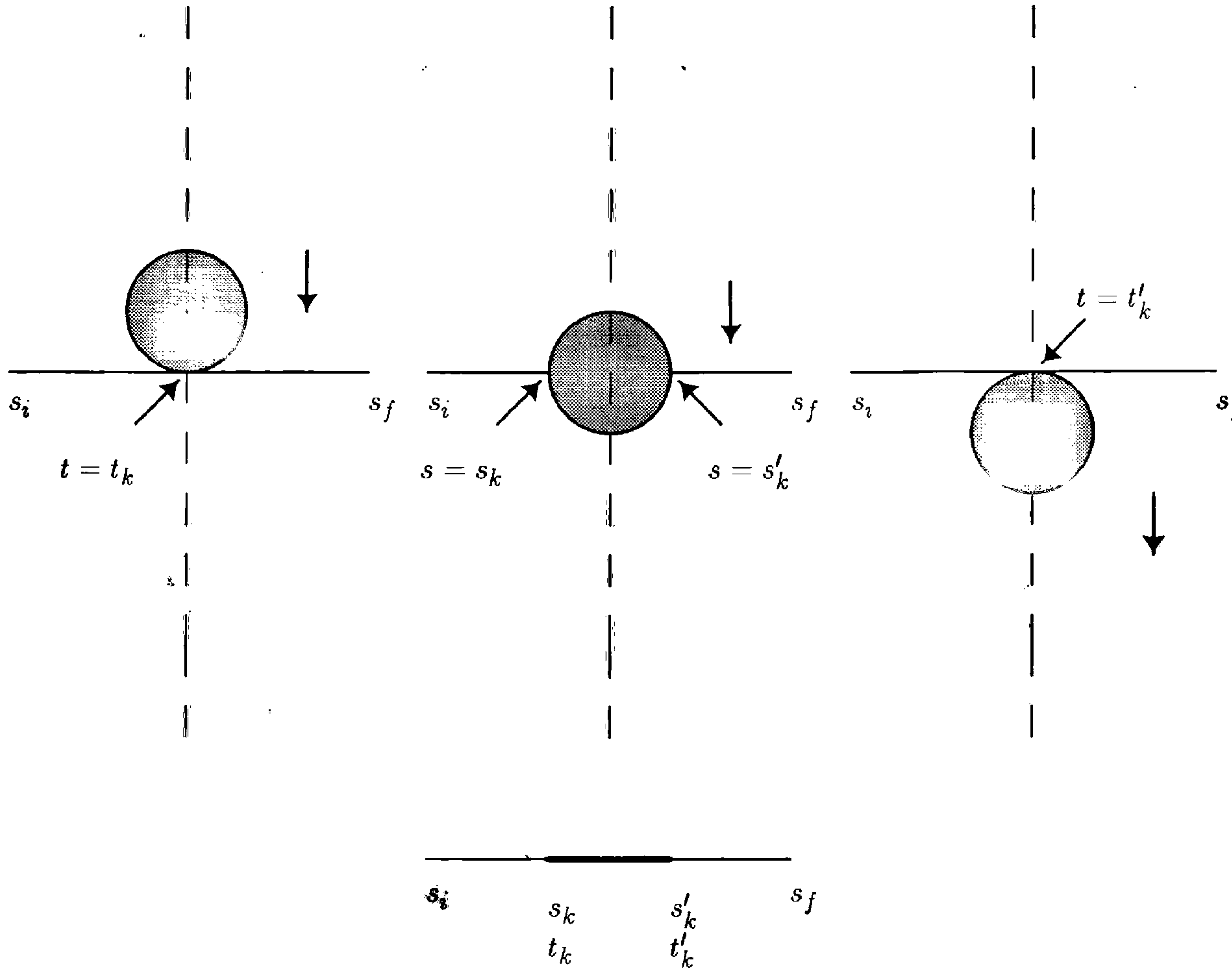
3.2. The VPP—Determining a Velocity Function

The robot is constrained to move along a fixed path, which is obtained by solving the PPP among static obstacles. Our next problem is to find the velocity of the robot as a function of time such that collisions with the moving obstacles are avoided (VPP). See Fig. 3 for an instance of VPP.

How is the path π specified? In general, it may be any arbitrary space curve. We assume a parametric representation for π in terms of its arc length s; i.e., $x_\pi(s)$ is given. Let $[s_i, s_f]$ be the arc length interval of the path π with $x_\pi(s_i) = x_i$, and $x_\pi(s_f) = x_f$. To find the mapping $x_\pi(\cdot)$, we need to find the mapping $s(\cdot): [t_i, t_f] \rightarrow [s_i, s_f]$. Then, by composition of mappings, $x_\pi(t) = x_\pi \circ s$ is easily determined.

* Note that two tetrahedra may be adjacent in three ways: they can share a vertex, an edge, or a face. The first two are singular cases, since a slight perturbation will disconnect the two tetrahedra. Therefore, the only stable adjacency is face adjacency.

Fig. 4. Computation of constraints for the velocity function. The obstacle is a circle. It is shown crossing the path segment [s_i, s_f]. *At* $t = t_k$, *it touches the path, and at* $t' = t'_k$, *it leaves the path. The maximum subsegment of the path occupied by the circle is* [s_k, s'_k] = d, *the diameter of the circle. This subsegment,* [s_k, s'_k] *is considered forbidden, since it is occupied by the obstacle, during the interval* [t_k, t'_k].



3.2.1. Path-Time ($s \times t$) *Space*

An object moving in space sweeps out an effective hyper-volume in space-time. An intuitive solution to the VPP can be obtained by intersecting the swept volumes of the obstacles with the given path of the robot. These intersections will provide time-varying constraints for the robot's position on the path. For simplicity and intuition, we shall illustrate the computation of the intersections for a simple case.

Consider the following two-dimensional case: the obstacle is a circle moving with constant velocity, and the path of the robot is a straight line segment. Let t_k be the time instant when the k^{th} obstacle touches the path segment, and let t'_k be the time instant when it leaves the segment. Given the velocity and the position of the circle (its center), t_k and t'_k are easily determined. During the whole interval [t_k, t'_k], the maximum subsegment of the fixed path $x_\pi(s)$ occupied by the obstacle is given by [s_k, s'_k] = d, the diameter of the circle.

As an approximation, these intersections may be taken as a subsegment of the path that is unavailable during a subinterval of time. Once these intersections have been determined, they act as constraints on the mapping $x_\pi(\cdot)$. The mapping may not assign any point t in subinterval [t_k, t'_k] to any point s in spatial subsegment [s_k, s'_k]. This is illustrated in Fig. 4.

Fig. 5. The subsegment, $[s_k, s'_k]$, of the segment $[s_i, s_f]$, is forbidden during the time interval $[t_k, t'_k]$. This is shown in the s × t space as a rectangle, which is a forbidden region for the trajectory to pass through.

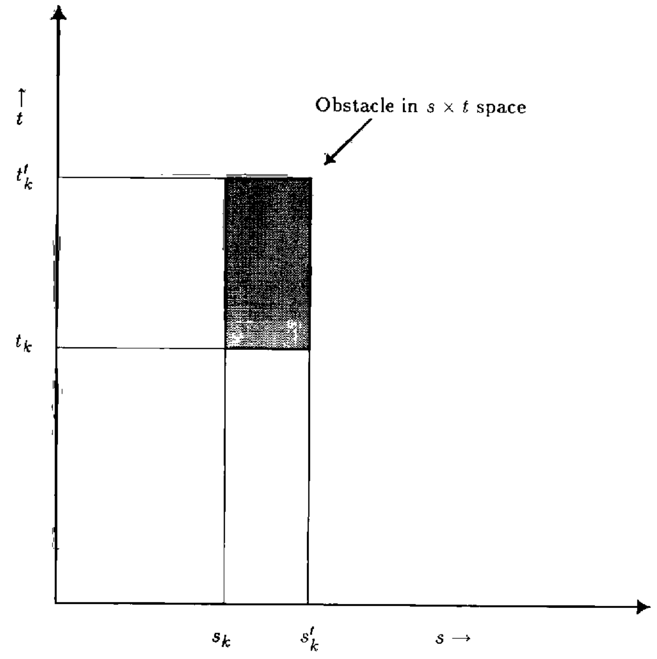
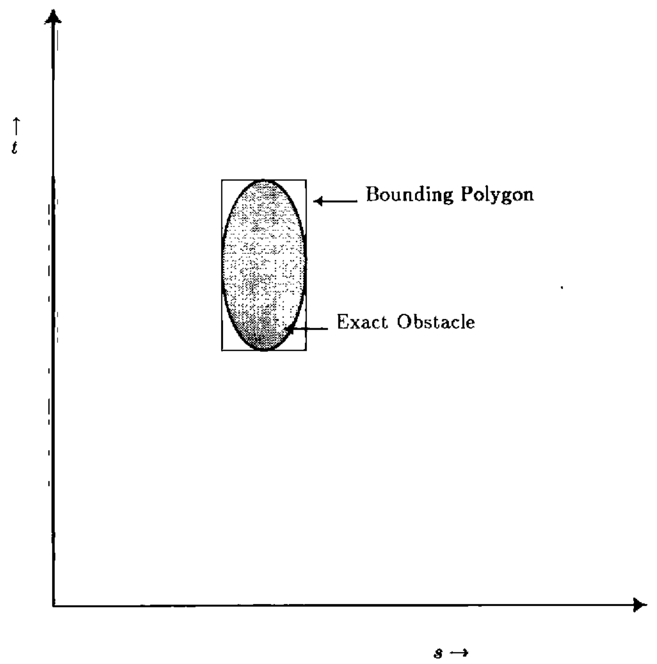


Fig. 6. A more exact computation of the swept volumes can give rise to more complicated shapes in the s × t space, e.g., an ellipse in the case of a circle crossing the path segment. For simplicity, bounding rectangular approximation to these shapes may be used.



Now consider $s \times t$ space. Since our approximations to the intersections are of the form—subsegment $[s_k, s'_k]$ is forbidden during the subinterval $[t_k, t'_k]$—they will appear as rectangles in $s \times t$ space. This is shown in Fig. 5.

Note that an exact computation of the volumes swept by the moving obstacles will give rise to more general shapes than rectangles (in $s \times t$ space), e.g., an ellipse for the circular obstacle moving with constant velocity, as illustrated in Fig. 6. It is a nontrivial task to compute the intersection of the path π with the volumes swept by the obstacles as they move around. Let $O^m = \{O_1^m, O_2^m, \ldots, O_r^m\}$, where O_k^m is an obstacle. Mathematically, we have to determine $\pi \cap O_k^m(t)$, $\forall t \in [t_i, t_f]$, $k = 1 \ldots r$. Therefore, these intersections will be quite complicated functions of time depending upon:

1. the shape and size of the obstacle,
2. the trajectory of the obstacle, and
3. the path π of the robot.

In Appendix I, we show that for a spherical obstacle moving with constant velocity, the exact intersection (in $s \times t$ space) is an ellipse; for a polyhedral obstacle moving with constant velocity, the exact intersection is a polygon. Here, a trade-off exists between precision and computation. In general, the more accurately a forbidden region is described, the more complex the intersection computation becomes.

For simplicity, we shall assume polygonal approximations to these intersections. Our results, however, may be extended to deal with more general shapes (in $s \times t$ space), such as ellipses, etc.

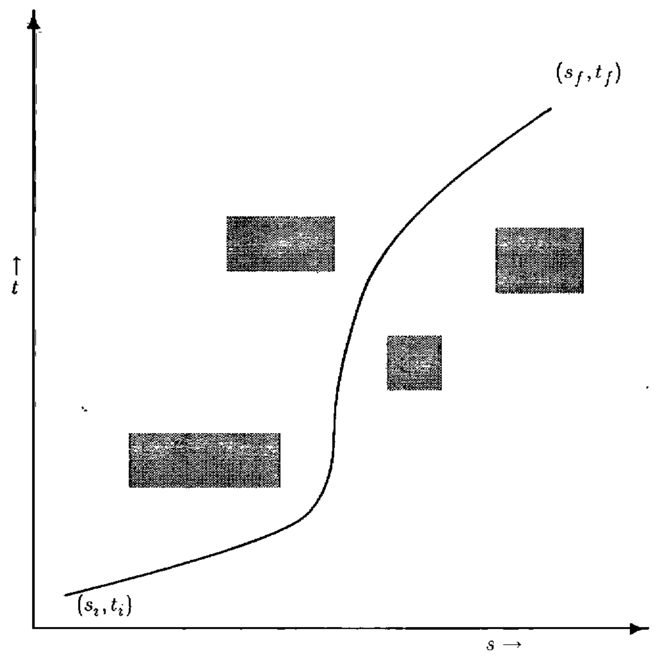
3.2.2. A Transformation: the VPP → the 2-D PPP in s × t Space

In $s \times t$ space, the mapping $s(t)$ will be a curve. A sufficient condition for no collision is that the intersection of the curve $s(t)$ with the obstacles (in $s \times t$ space) be null. Thus, we have reduced the VPP to the following problem:

> Given the initial point (s_i, t_i) and the final point (s_f, t_f), find a curve in $s \times t$ space joining these two points, while avoiding the (polygonal) obstacles (see Fig. 7).

In an abstract form, this is precisely equivalent to the PPP in $s \times t$ space. Several techniques are available for solving it (see Section 3.1). While we are developing

Fig. 7. Equivalence of the velocity planning problem to the static two-dimensional path planning problem in the s × t *space. The horizontal axis is the arc length and the vertical axis is the time. The shaded rectangular regions (computed as in Figs. 4 and 5) represent the two-dimensional obstacles through which the trajectory may not pass.*



approaches based on free-space representations, in the next section we will present an approach based on the Vgraph algorithm. We extend it to solve the two-dimensional PPP (in $s \times t$ space) corresponding to the VPP. Our purpose is to illustrate the differences between path planning in spatial dimensions and path planning in space-time ($s \times t$ space). Note that the end conditions matter. The first case, which leads to the VPP algorithm 1, assumes that the final arrival time t_f is fixed. The second case, which leads to the VPP algorithm 2, assumes that t_f is not fixed but that the objective is to reach the goal in minimum time.

3.2.3. *Constraints in s × t Space*

At first glance, it appears that we may use the Vgraph algorithm to solve the VPP, but this is not so. In $s \times t$ space, certain additional constraints exist because one of the dimensions is time. These constraints are:

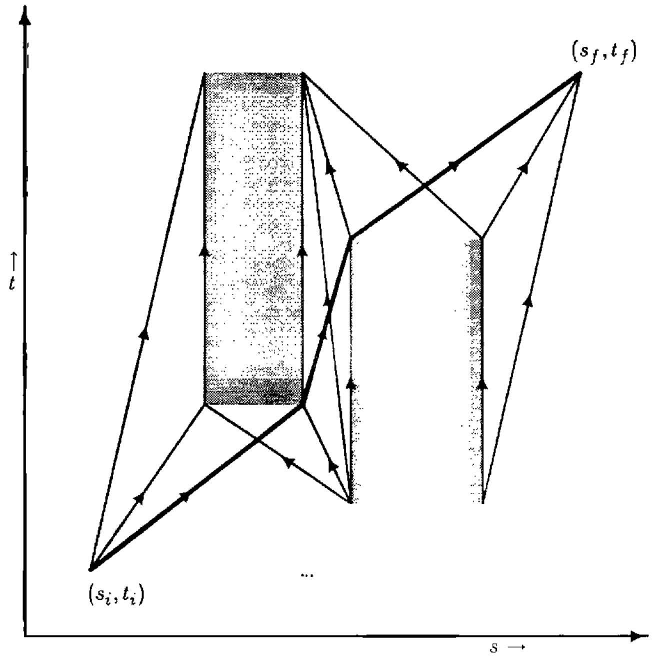
1. The monotonicity constraint. Time moves forward! It implies that any physically realizable function in $s \times t$ space has to be a monotonically increasing function of t.
2. The maximum velocity constraint. The velocity of the robot is bounded, i.e., $|v(t)| \leq V_{max}$. What does this constraint imply in $s \times t$ space? The velocity of the robot is $v(t) = \frac{dx_\pi}{dt} = \frac{dx_\pi}{ds}\frac{ds}{dt}$. Since $x_\pi(s)$ is given, $\frac{dx_\pi}{ds}$ is known. Hence, the constraint on $\frac{dx_\pi}{dt}$ may be transformed to a constraint on the slope $\frac{ds}{dt}$ in $s \times t$ space. Since s represents arc length, $\left|\frac{dx_\pi}{ds}\right| = 1$.* Therefore, $\left|\frac{dx_\pi}{dt}\right| = \left|\frac{ds}{dt}\right| \leq V_{max}$. The same V_{max} constraint holds for the slope $\left|\frac{ds}{dt}\right|$. Note that the time axis is the vertical axis. Therefore, the slope corresponds to the cotangent and not the tangent, which is more usual.

3.2.4. *DVG: Vgraph in s × t Space*

The Vgraph algorithm for the two-dimensional PPP cannot be used identically in $s \times t$ space, but requires two modifications. First, the monotonicity constraint makes the Vgraph in $s \times t$ space a directed graph, since a node (s_j, t_j) may be accessible from another node (s_k, t_k) only if $t_k < t_j$. Second, the maximum velocity constraint translates to a constraint on the slopes of the edges in the Vgraph. The edges with $\left|\frac{ds}{dt}\right| > V_{max}$ are not admissible. These modifications can be formally incorporated in a directed visibility graph, $DVG(N, E)$, as follows: Let O be the set of polygonal forbidden regions in $s \times t$ space. Let V be

* In case s is not the arc length parameter, let $\left|\frac{dx_\pi}{ds}\right| = \gamma$. γ will in general depend on s. The velocity constraint now translates to $\left|\frac{ds}{dt}\right| \leq V^2_{max} = V_{max}/\gamma$. The maximum velocity constraint still translates to a slope constraint, but the constraint depends on s. A conservative constant constraint may be obtained by using the maximum value of γ.

Fig. 8. The constraints in s × t *space modify the visibility graph to a directed graph DVG. The edges corresponding to velocity greater than* V$_{max}$ *are pruned. A minimum path search over the pruned graph gives an optimal path in the* s × t *space. The minimum path from* I *to* F *is shown in bold lines.*



the set of vertices of these forbidden regions. Construct $DVG(N, E)$ as follows. The node set $N = \{I, F\} \cup V$. To determine the edge set E, first define a *cone-visibility* function l for a node pair (n_i, n_j). Let the coordinates of node $n_i = (s_i, t_i)$ and those of node $n_j = (s_j, t_j)$. Then, $l(n_i, n_j)$ is TRUE if:

1. $\overline{n_i, n_j} \cap O = \phi$,
2. $\left|\frac{s_j - s_i}{t_j - t_i}\right| \leq V_{max}$, and
3. $t_i < t_j$.

The edge set E is then defined as the set of all edges (n_i, n_j) such that $l(n_i, n_j)$ is true. The graph DVG for the VPP in Fig. 3 is shown in Fig. 8.

3.2.5. VPP Algorithm 1: Fixed Arrival Time

Is an optimal path still composed of segments joining the vertices of the obstacles, or do we have to introduce new vertices corresponding to $\left|\frac{ds}{dt}\right| = V_{max}$? The answer is stated in the following theorem (see Appendix II for proof):

The MinPath Theorem in s × t *space with slope constraint:* The minimum length collision-free path (in $s \times t$ space), from initial point I to final point F and satisfying the slope constraint, is composed of directed straight line segments $\left(\left|\frac{ds}{dt}\right| \leq V_{max}\right)$ joining vertices of the polygonal forbidden regions.

As a consequence of this optimality, the following corollary follows.

Corollary: The directed graph DVG is complete. If a (graph) path from node I to node F does not exist, this implies that a collision-free velocity profile to the goal node does not exist.

Note that the Euclidean distance between two nodes (s_j, t_j) and (s_k, t_k) in $s \times t$ space corresponds to $\sqrt{(s_k - s_j)^2 + (t_k - t_j)^2}$. This is equivalent to $\sqrt{(1 + v^2)}|t_k - t_j|$ where $v = \frac{s_k - s_j}{t_k - t_j}$. The *optimal* path corresponds to a velocity profile that *minimizes* the above cost function over all possible velocity profiles. This cost function is related to the energy dissipated, but it does not account for switching from one velocity value to another, an important component of total expended energy. However, this optimal property does ensure the completeness of the algorithm.

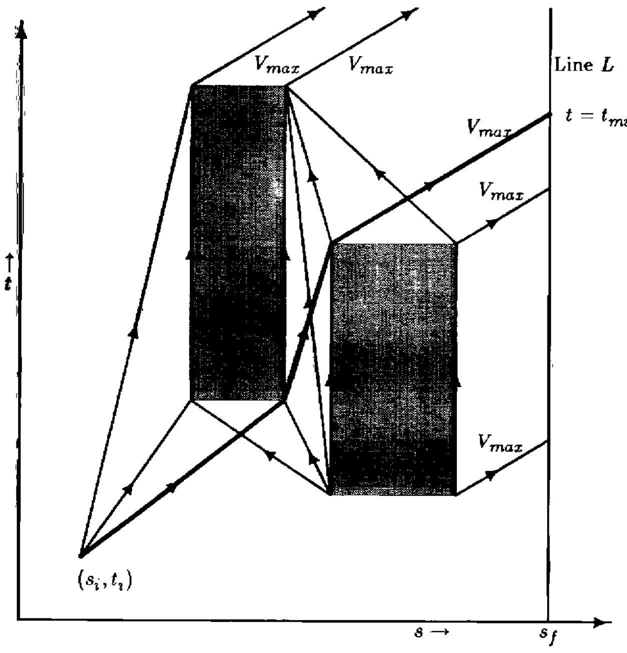
Given the set of polygonal obstacles O in $s \times t$ space, initial node I, and final node F, the VPP algorithm 1 (fixed arrival time) may be summarized as follows:

1. Construct the directed visibility graph, *DVG*.
2. Search the graph for a path from I to F.
3. *If* the path exists, *then* it corresponds to a collision-free velocity profile; else, a collision-free velocity profile does not exist.

3.2.6. VPP Algorithm 2: Minimum Time

Another version of the VPP is one in which the final position s_f is specified and the aim is to reach it in minimum time t_{min}. The problem is less constrained now. Instead of a given goal node, a set of goal nodes is given. This set is the line L drawn from (s_f, t_i) parallel to the t-axis as shown in Fig. 9. Let t_{min} be the minimum time required to reach the final position s_f. The node (s_f, t_{min}) has to lie somewhere on the vertical line

Fig. 9. The final position s$_f$ *is specified in this case. The time when the robot reaches* s$_f$ *is to be minimized. It is determined by the intersection of a trajectory, one of many possible ones, with the vertical line* L *that passes through* s$_f$. *The* V$_{max}$ *(and* −V$_{max}$*) probes intersecting the line* L *give rise to potential final nodes. We have shown only the* V$_{max}$ *probes. If a probe intersects an obstacle before it intersects the line* L*, no potential goal node is generated. Minimum time corresponds to the node that is reachable and has minimum* t *coordinate. The minimum time path is shown in bold lines.*



L, which passes through the final position s_f. But the number of points on L is infinite. The following lemma (see Appendix II for proof) shows that not all points on line L may be the final node (s_f, t_{min}).

> *The MinTime-node Lemma in* s × t *space with slope constraint:* The minimum time node (s_f, t_{min}) is an intersection point of line L and a line segment of slope V_{max} (or $-V_{max}$) emanating from an obstacle vertex.

We need to modify the graph DVG to a new graph DVG' as follows. For each node in DVG, as obtained in the previous section, construct probes in the increasing t direction (a ray) with slopes corresponding to (V_{max} and $-V_{max}$). If a probe intersects the line L without intersecting any obstacle first, i.e., if this probe can "see" the line L, the point of intersection will be a new potential final node (see Fig. 9).

Let the set of these potential final nodes be N_f. DVG is augmented with these new potential final nodes. The new node set becomes $N' = N \cup N_f$. The edge set is augmented with edges to nodes in N_f. Let $E_f = \{(n_i, n_j) : l(n_i, n_j)$ is TRUE, $\forall n_i \in N, \forall n_j \in N_f)\}$. The new edge set is $E' = E \cup E_f$.

Instead of one final node, we have a set N_f of final potential nodes. The node in set N_f, which may be reached from the initial node I and which has the minimum time coordinate of all the nodes (in N_f) reachable from I, is the required goal node; the corresponding time is the minimum time solution to reach the final position s_f. The velocity profile corresponding to the path to this node is therefore optimal. The minimum time node may be determined by searching the graph with respect to time as a cost function, i.e., the cost assigned to an edge (n_i, n_j) is the difference in time coordinates of the nodes $(t_j - t_i)$.

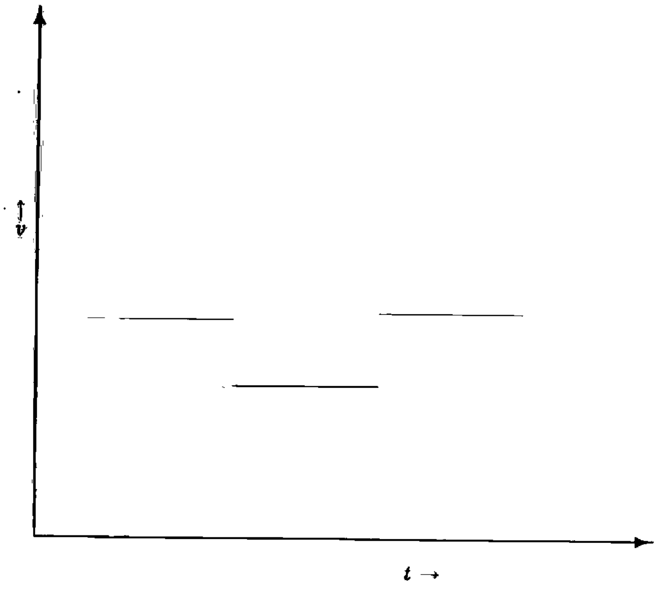
We can summarize the above as follows: the minimum time collision-free path (in $s \times t$ space) from initial point I to final position s_f (which corresponds to the set N_f of potentially final nodes), satisfying the slope constraint, is composed of directed straight line segments $\left(\left|\frac{ds}{dt}\right| \leq V_{max}\right)$ joining vertices of the polygonal forbidden regions. Thus, the VPP algorithm 2 (minimum time) can be summed up as follows:

1. Construct the graph DVG'.
2. Search for the minimum (time) cost path from node I to nodes in set N_f.
3. *If* the minimum cost path exists, *then* it corresponds to the minimum time collision-free velocity profile *else* a collision-free velocity profile does not exist.

3.2.7. Velocity Profile—Smoothness Requirements

Consider Fig. 8. The optimal path (from node I to node F) in $s \times t$ space corresponds to the velocity function shown in Fig. 10. The velocity profile is discontinuous, implying infinite acceleration. This is clearly unrealizable physically and occurs because straight line segments were used to connect the vertices. It results in C^0 continuity. To keep acceleration finite, we require at least C^1 continuity. This can be obtained by smoothly interpolating between the vertices, (through which the path in $s \times t$ space passes), with quadratic splines (De Boor 1978). Care must be taken to guarantee that the interpolated function does not intersect the forbidden constraint regions. We

Fig. 10. The velocity profile corresponding to the path in the s × t *space in Fig. 8. Note that the velocity profile is discontinuous, thereby implying infinite acceleration. A smooth quadratic spline fit is required. But the spline fit is constrained by the obstacles in the* s × t *space, i.e., it may not intersect the obstacles.*



have now developed an approach based on spline interpolation in free space (see Section 3.1.2) to obtain a smooth velocity profile that lies completely in free space (Kant and Zucker 1986).

3.3. Summary

We have presented algorithms to solve the TPP $\Rightarrow$ PPP + VPP. We used the triangulation approach to find a path that avoids collision with fixed obstacles to reach the goal position. Then we presented the VPP algorithm to plan the velocity of the robot along this path to avoid collisions with moving obstacles.

As we show in the next section, this heuristic decomposition results in a significant reduction in the complexity of the problem. The trajectory yielded by this decomposition may not always be good, for it may take too long to reach the goal position if there are too many obstacles crossing the chosen path. This decomposition may also fail if the obstacle is coming at the robot on the same path or if an obstacle is tracking the robot. The robot may have to move off the path to avoid the tracking obstacle.* Therefore, our approach will work best for obstacles that are sparsely distributed and moving relatively orthogonally to the path of the robot.

* We have now developed an algorithm that allows the robot to move off the path while keeping the velocity magnitude constant. This will be reported elsewhere.

4. Complexity Issues

What do we gain by the decomposition approach to solving the TPP? The answer is that we reduce the complexity of the problem. The time complexity of one algorithm (Schwartz and Sharir 1982*a*) to solve a fairly general version of the PPP is $O((2n)^{3^{r+1}}m^{2^r})$, where m and n are related to the number and the maximum degree of polynomials describing the obstacles and r is related to the number of degrees of freedom of the robot. The complexity is exponential in the number of degrees of freedom of the robot and is polynomial in the number of geometric constraints; i.e., the number and degree of polynomial equations describing the obstacles. However, the exponents and constants are large from a practical point of view.

The full problem of planning trajectories among moving obstacles is even harder. Recently Reif and Sharir (1985) have shown that some TPPs are NP-hard (e.g., moving a disc in 3-D space with static and rotating obstacles in minimal time). One reason for this increase in complexity is the additional time dimension in the TPP. We have already noted that the complexity of the PPP is exponential in the number of degrees of freedom, which for a *point* robot is the same as the dimension of the space in which the robot is embedded. For instance, in 3-D space, a point robot has three degrees of freedom.

The TPP for a point robot could be formulated as a problem in four-dimensional $x \times y \times z \times t$ space-time. Our approach, PPP + VPP, reduces this four-dimensional problem to a problem in 3-D $x \times y \times z$ physical space, followed by a problem in two-dimensional path-time space, thereby reducing the time complexity greatly.

A point robot is an unrealistic assumption; any physical robot will have nonzero size. How do we take into account the size and shape of the robot? An elegant approach, called the *Cspace* approach, has been developed by Lozano-Perez (1980). In this approach, the robot is "shrunk" to a point while the obstacles are "grown". Moving this virtual point robot among "grown obstacles" is equivalent to the original problem. The problem is transformed to an equivalent problem in the space (*Cspace*) of possible configurations (position and orientation) of the robot. In this space, each point represents a configuration of the robot. Thus, solving a TPP for an r degree-of-freedom robot may be formulated as a problem in $r + 1$ dimensional *space-time.* We could use our decomposition approach to solve it in two steps: (1) solve the PPP in n-dimensional *Cspace,* and (2) solve the VPP in two-dimensional *path-time* space. The computation involved in transforming the obstacles to *Cspace,* and searching this high dimensional *Cspace* is expensive and complex, however. Despite the complexity of the resulting PPP, the decomposition approach still reduces the overall complexity of the TPP. Unfortunately, under some situations it may not yield a solution even though one may exist.

The decomposition approach may also be extended to approximate but fast algorithms for solving the TPP, e.g., the case of a polygonal robot in a two-dimensional polygonal world. The generalized cones approach by Brooks (1982) may be used to solve the PPP, and the VPP algorithm (see Sections 3.2.5 and 3.2.6) may be extended to plan only the translational velocity of the polygonal robot to avoid collision with the moving obstacles. The same decomposition approach may be used for a host of other TPPs in time-varying environments. For instance, it has been applied to a simple case of two manipulators (in plane) sharing a common workspace (Wallace 1984).

5. Discussion and Conclusion

We have considered the problem of planning trajectories in time-varying environments. Other approaches have been restricted to stationary environments within which general solutions, while possible in theory, are computationally expensive in practice. Therefore, planning trajectories among moving obstacles will be even harder.

We have presented a paradigm that suggests an opening through this increased complexity. The essential feature of this paradigm is to decompose the full problem of planning trajectories into two subproblems: (1) planning the path of the robot so collision with static obstacles is avoided; and (2) planning the velocity of the robot along this path so collisions with the moving obstacles are avoided. This heuristic decomposition results in a significant reduction in the complexity of the problem. Under certain conditions, it still may not give a solution even though one may exist.

Our solution assumes that the obstacles and their trajectories are completely known. The robot computes the path and the velocity profile in one global action. Having obtained this information, the robot moves "blindly" along the path with the velocity. However, the global plan may not be executed "blindly" since the obstacles may not move as predicted. Often, the exact trajectories of the moving obstacles may not be known, in which case the trajectories of these obstacles have to be estimated repeatedly and the trajectory of the robot planned repeatedly. For example, the decomposition paradigm may be applied locally by planning the path locally in a series of actions and by planning the velocity over subintervals of the total "task" time. The size of these intervals, or, the "granularity" of time will then be proportional to the confidences associated with the statistical estimates of the movements of the obstacles. If the movements of the obstacles are predictable, few instances of VPP over large "chunks" of time need to be solved, or velocity needs to be planned more "globally" in time. If the movements of the obstacles are almost random, VPP may have to be solved almost continuously over small "chunks" of time, or velocity may need to be planned more locally in time. The granularity of time is linked to the predictability of the movements of the obstacles. We are currently developing approaches to trajectory planning problems that incorporate this and that attempt to resolve some of the *local/global* planning issues involved in planning trajectories.

Acknowledgments

We wish to thank Hossam El Guidy, Yvan Leclerc, and Susan Whitesides for discussions and help with this document.

Appendix I. Computing Forbidden Regions in $s \times t$ Space

I.A. General Procedure

We assume that the obstacles are represented by their boundaries. In n dimensional Euclidean space E^n, let the obstacle boundaries be represented by $f_i(\vec{r}) = 0$, $i = 1, m$ where $\vec{r}$ is the position vector. This may be represented as $\mathbf{f}(\vec{r}) = 0$ where $\mathbf{f}$ is a vector with f_i as its elements. Let the fixed path be an n-dimensional curve segment π parameterized by s, $\pi = \pi(s)$, $s \in [s_i, s_f]$. Let the motion of the obstacle be represented by a Euclidean matrix transformation $\mathbf{T}(t)$, defined as a matrix function of time t. The forbidden region in $s \times t$ space is given by $\mathbf{T}(t)\mathbf{f}(\vec{r}) = 0$, $\vec{r} = \pi(s)$.

The above set of simultaneous, nonlinear (in general) equations will determine the forbidden region. Solving these equations may require sophisticated algebraic techniques. For some cases (pure translation with constant velocity, no rotation), the computations are quite simple and easy. We illustrate two cases below. In both cases, the path is a straight line segment. In the first case, the obstacle is a sphere moving with constant translational velocity; in the second case, the obstacle is a convex polyhedron moving with constant translational velocity. We shall compute the intersection with the infinite line containing the path segment. A part, or all, of this intersection may lie outside the path segment. This is easily determined by simple limit checks.

I.B. Spherical Obstacle

Let the path π be a straight line segment, characterized parametrically by

$$\frac{x - x_0}{a} = \frac{y - y_0}{b} = \frac{z - z_0}{c} = s, \qquad s \in [0, l]. \quad (1)$$

Let the position of (the center of) the spherical obstacle at $t = 0$ be (x_{c0}, y_{c0}, z_{c0}). Let its velocity be $v = (v_x, v_y, v_z)$ and its radius r. The equation of the volume swept by the spherical obstacle as a function of time is given by

$$[x - (x_{c0} + v_x t)]^2 + [y - (y_{c0}0 + v_y t)]^2 + [z - (z_{c0} + v_z t)]^2 = r^2. \quad (2)$$

Substituting for x, y, and z from Eq. (1) in terms of s, and rearranging, we get

$$[(x_0 - x_{c0}) + as - v_x t]^2 + [(y_0 - y_{c0}) + as - v_y t]^2 + [(z_0 - z_{c0}) + as - v_z t]^2 = r^2. \quad (3)$$

Let $\Delta x = x_0 - x_{c0}$. From Eq. (3), we get

$$\begin{aligned} s^2(a^2 + b^2 + c^2) &- 2st(av_x + bv_y + cv_z) \\ &+ t^2(v_x^2 + v_y^2 + v_z^2) + 2s(a\,\Delta x + b\,\Delta y + c\,\Delta z) \\ &- 2t(v_x\,\Delta x + v_y\,\Delta y + v_z\,\Delta z) \\ &+ (\Delta x^2 + \Delta y^2 + \Delta z^2 - r^2) = 0. \end{aligned} \quad (4)$$

This is a quadratic in s and t. Therefore, the forbidden region, if it exists, will be a conic section. Moreover, a general conic in the xy plane, $Ax^2 + Bxy + Cx^2 + Dx + Ey + F = 0$, is a hyperbola, parabola, or an ellipse as the discriminant $B^2 - 4AC$ is greater than, equal to, or less than 0. From Eq. (4), the discriminant is

$$4(av_x + bv_y + cv_z)^2 - 4(a^2 + b^2 + c^2)(v_x^2 + v_y^2 + v_z^2). \quad (5)$$

By expanding and rearranging terms, Eq. (5) is reduced to

$$-[(av_y + bv_x)^2 + (bv_z + cv_y)^2 + (cv_x + av_z)^2], \quad (6)$$

which, being the negative of the sum of the squares, is always less than or equal to zero. It can be shown that Eq. (6) is always less than zero. Hence, the intersection in $s \times t$ space will be an ellipse. The ellipse, in some special cases, may be degenerate.

I.C. Convex Polyhedral Obstacle

The path π is again given by Eq. (1). The obstacle is a convex polyhedron moving with constant translational velocity $v = (v_x, v_y, v_z)$. For intuitive simplicity, we may think of the problem as the line segment moving with velocity $-v$ and the polyhedron as stationary. The forbidden region in $s \times t$ space is the intersection of the polyhedron with the plane swept by the line segment. This plane is easily characterized by

$$x = x_0 + as - v_x t,\ y = y_0 + bs - v_y t, \\ z = z_0 + cs - v_z t. \tag{7}$$

The forbidden region is determined by intersecting the plane with the boundaries of the faces of the polyhedra. Since a face is a convex polygon, its intersection with the swept plane will be a line segment with its two vertices on the boundary edge segments of the face. Any two line segments will share a vertex if the two faces share an edge. The forbidden region, a polygon enclosed by these line segments, is easily determined.

Let the equation of the i^{th} boundary edge segment of a face be given by

$$\frac{x - x_i}{p_i} = \frac{y - y_i}{q_i} = \frac{z - z_i}{r_i} = \lambda \qquad 0 \leq \lambda \leq 1, \tag{8}$$

where (x_i, y_i, z_i) is one of the vertices.

Substituting for x, y, z from Eqs. (7) and (8), we get

$$p_i\lambda - as + v_x t + (x_i - x_0) = 0,$$

$$q_i\lambda - bs + v_y t + (y_i - y_0) = 0,$$

$$r_i\lambda - cs + v_z t + (z_i - z_0) = 0.$$

We can easily solve this set of linear equations for s, λ, t. If $0 \leq \lambda \leq 1$, it implies a valid intersection point. Knowing these points, the polygonal forbidden region $s \times t$ space is easily determined.

Appendix II. Proofs for some Vgraph-like Algorithms

Intuitively, the two-dimensional PPP in a plane may be stated as the problem of moving a point object from the initial position I to the final position F while avoiding the polygonal obstacles.* Any shortest (optimal) path satisfies the following property: An *optimal collision-free* path from the initial point I to the final point F is composed of straight line segments joining vertices of the obstacles.

A proof based on dynamic programming is given in Wangdahl, Pollock, and Woodward (1974). We will present an intuitive proof in the following section. Next, we show that a similar theorem holds in $s \times t$ space even with slope constraint. Finally, a lemma for the minimum time path in $s \times t$ space is proved. Our proofs are based on the following lemma:

> ***Lemma 1.*** The length of a *convex* segment ψ, between two points P and Q is greater than any convex segment ψ' that lies completely inside the convex polygon formed by ψ and the segment $\overline{PQ}$.

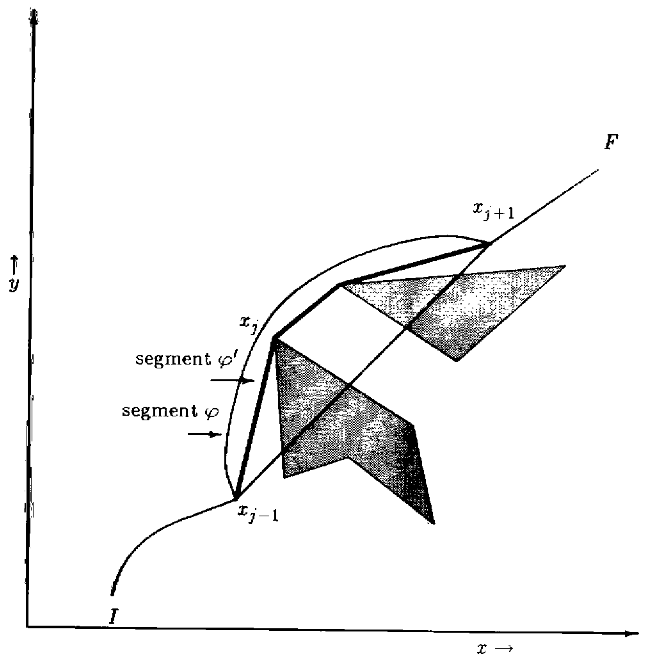
II.A. MinPath Theorem

> *Given:* Initial point I, final point F, and set O^s of a finite number of static polygonal obstacles.
>
> *To prove:* An *optimal* path ψ from the initial point I to the final point F, such that $\psi \cap O^s = \emptyset$ is composed of straight line segments joining vertices of the obstacles.
>
> *Proof:* We shall prove the theorem by contradiction, i.e., we shall assume that a point on an optimal path does not satisfy the above condition, and we will show that a path composed of straight line segments joining vertices of the obstacles exists that is shorter than the given optimal path. While following this proof, refer to Fig. A.1.

Let V be the set of vertices of the obstacles, the initial point I, and the final point F. Let E be the set of edges v_iv_j, such that v_i, $v_j \in V$ and $\overline{v_iv_j}$ does not intersect any of the obstacles. Let x_j be a point on the optimal path ψ such that x_j does not satisfy the above property, i.e., $x_j \notin V$ and $x_j \notin e$, where $e \in E$. Let x_{j-1}, x_j and x_{j+1} be the largest convex subsegment φ,

* The obstacles are open sets, i.e., a collision-free path may touch the boundary of the obstacles.

Fig. A.1. *Illustration for the proof of the MinPath Theorem.*



of the given optimal path ψ, such that x_{j-1} lies in between I and x_j, and x_{j+1} lies in between x_j and F.

Claim 1. x_{j-1}, x_j, and x_{j+1} do not lie on a straight line. This follows from the fact that these points form the largest convex subsegment of the whole path φ.

Claim 2. The subsegment φ is minimal, i.e., it is a shortest path between x_{j-1}, and x_{j+1}. This follows from the *principle of optimality*. Since the whole path ψ is minimal, any part of it is minimal also. But,

$$|\varphi| > |\overline{x_{j-1}x_{j+1}}|.$$

Therefore, $\overline{x_{j-1}x_{j+1}}$ must intersect an obstacle or φ will not be minimal. Also, φ does not intersect any obstacles since we assume it is collision-free.

> $\Rightarrow \exists$ a nonempty set V' of the vertices (of obstacles) such that $V' \subseteq V$ and $\forall\ v \in V'$, v lies inside the convex set S, formed by the straight line segment $\overline{x_{j-1}x_{j+1}}$, and the convex subpath, φ.

Let $x_{j-1}v_1 \ . \ . \ . \ v_rx_{j+1}$ be the vertices of the convex hull of $x_{j-1} \cup V' \cup x_{j+1}$. Consider the convex subsegment $\varphi' = x_{j-1}v_1 \ . \ . \ . \ v_rx_{j+1}$. φ' lies completely inside the convex set S. Therefore, by Lemma 1,

$$|\varphi'| < |\varphi|.$$

This is a contradiction since φ is a minimal path between x_{j-1} and x_{j+1}.

$$\Rightarrow x_j \notin \varphi.$$

II.B. MinPath Theorem in $s \times t$ Space with Slope Constraint

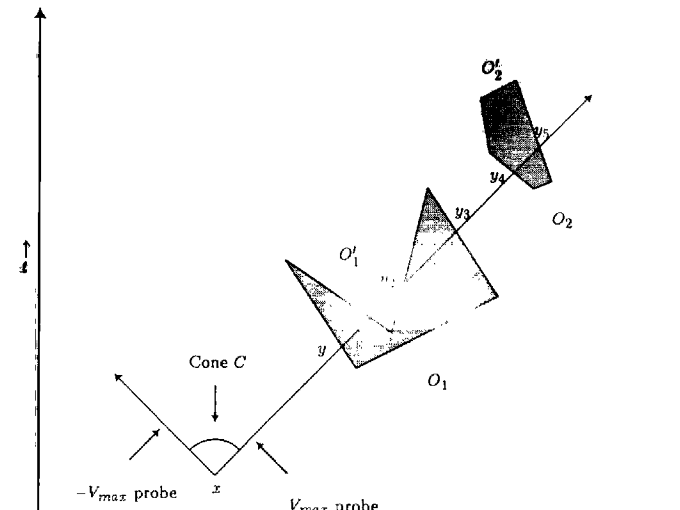
In $s \times t$ space there are additional constraints. In particular: (1) the path has to be monotonic in t, and (2) the slope is $\left|\frac{ds}{dt}\right| \le V_{max}$, where V_{max} is a positive constant.* We will prove that an optimal path (minimum length in $s \times t$ space) is still composed of straight line segments joining vertices of the obstacles.

> *Given:* The initial point I, the final point F, and the set O of finite number of polygonal obstacles.
>
> *The maximum slope constraint:* The magnitude of the slope of the path $\left|\frac{ds}{dt}\right|$ is always $\le V_{max}$.
>
> *To prove:* An optimal collision-free path from the initial point I to the final point F, satisfying the maximum slope constraint, would be composed of straight line segments (slope $\le V_{max}$) joining vertices of the obstacles.
>
> *Proof:* From the MinPath Theorem, we know that if there were no maximum slope constraint, an optimal path would be composed of the straight line segments joining the vertices of the obstacles. How does the maximum slope constraint affect the nature of this path? Consider an arbitrary point x. Geometrically, the constraint implies that any edge emanating from the point x is admissible if it lies within the conical region shown in Fig. A.2. The boundaries of this conical region C correspond to lines with slopes $\pm V_{max}$.

Formally, this is incorporated in the notion of *cone visibility*. A node y is cone visible from a node x if the following visibility predicate $I(x, y)$ is TRUE. Let the

* Note that the time axis is the vertical axis. Therefore, the slope corresponds to the cotangent, not the tangent, as is more usual.

Fig. A.2. Illustration for the proof of the MinPath Theorem in s × t *space with slope constraint.*



coordinates of node $x = (s_i, t_i)$ and those of node $y = (s_j, t_j)$. Then, $l(n_i, n_j)$ is TRUE if:

1. $\overline{n_i, n_j} \cap O = \emptyset$,
2. $\left|\dfrac{s_j - s_i}{t_j - t_i}\right| \le V_{max}$, and
3. $t_i < t_j$.

Essentially, the equivalent obstacles for point x are those parts of the obstacles that lie within the cone C. If all the obstacles lie completely inside the cone, we have nothing to prove. If an obstacle O_j intersects the cone, it is essentially transformed into a new obstacle $O_j' = O_j \cap C$. The new obstacle O_j' will also be a set of simple polygons, as shown in Fig. A.2. We have shown intersection with the $+V_{max}$ probe only. The case for $-V_{max}$ is similar. The new obstacle has extra nodes, $y, y_1, y_2, \ldots, y_r$. We need to consider only the first node y, since this is the only node (in this new set of y nodes, $y_1, y_2, \ldots, y_r$) that is cone-visible from point x. The other nodes will not even be reachable from x because of the obstacle, as shown in Fig. A.2.

Intuitively, we have transformed the original constrained problem with obstacles O to an unconstrained problem with obstacles O'. But the obstacles O' have extra nodes, e.g., y nodes, created by the maximum slope lines. If we apply the MinPath Theorem to obstacles O', we may have to consider these new nodes. We shall now prove that such is not the case, i.e., one need not create these extra nodes to find an optimal path. We have to show that any point z that is cone-visible from the new node y is reachable from x by a path φ with the following properties:

1. It is composed of straight line segments (slope $\le V_{max}$) joining vertices of the obstacles.
2. Its length is $|\varphi| \le |\overline{xy}| + |\overline{yz}|$.

But we will prove a stronger result. We will show that any point z that is cone-visible from the new node y is reachable from any point x', from which y is cone-visible by a path φ with the above properties.

The set of points from which the node y is cone-visible lies in the inverted cone bounded by $\overline{xy}$ and the edge of the obstacle that gave rise to y. Let x' be a point from which node y is cone-visible. Consider the straight line segment $\overline{x'z}$. Since z lies in the visibility cone of point y, slope $(\overline{yz}) \le V_{max}$. Also, the visibility cone of $y \subseteq$ visibility cone of x', $\Rightarrow$ slope$(\overline{x'z}) \le V_{max}$. There are two possibilities:

1. $\overline{x'z}$ does not intersect any obstacle, or
2. $\overline{x'z}$ does intersect some obstacles.

In the first case, $\overline{x'z}$ is a valid edge segment (slope $\le V_{max}$) and thus is a valid path. Moreover, $|\overline{x'z}| \le |\overline{x'y}| + |\overline{yz}|$.

In the second case, $\exists$ a set of vertices of obstacles inside the $\Delta x'yz$, is shown in Fig. A.3. Obstacles intersect $\overline{x'z}$ but not $\overline{x'y}$ or $\overline{yz}$ since we assume these are valid edges. Let this set of vertices be V'. Consider the convex hull of $x' \cup V' \cup z$. Let the vertices of this convex hull be $x', v_1, v_2, \ldots, v_r, z$. The path segment, $\varphi = x'v_1v_2 \ldots v_rz$, is a valid segment, since all the edge segments of this path have slopes $\le V_{max}$. This is because:

1. The path segment φ is a convex segment, (part of the convex hull).
2. The path segment φ lies inside the $\Delta x'yz$, each of whose sides, $\overline{x'y}$, $\overline{yz}$, and $\overline{x'z}$, have slope $\le V_{max}$.

Moreover, by Lemma 1, $|\varphi| \le |\overline{x'y}| + |\overline{y'z}|$. Hence, any point cone-visible from node y is reachable from any

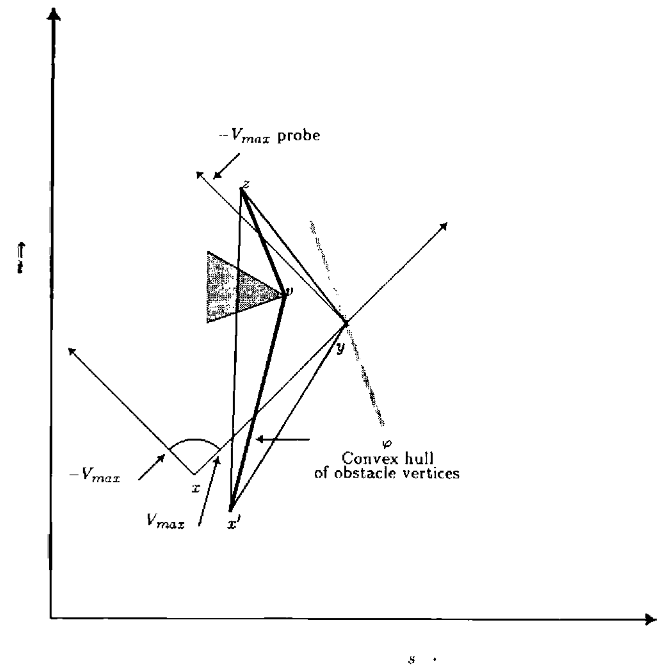


Fig. A.3. Illustration for the proof of the MinPath Theorem in s × t *space with slope constraint.*

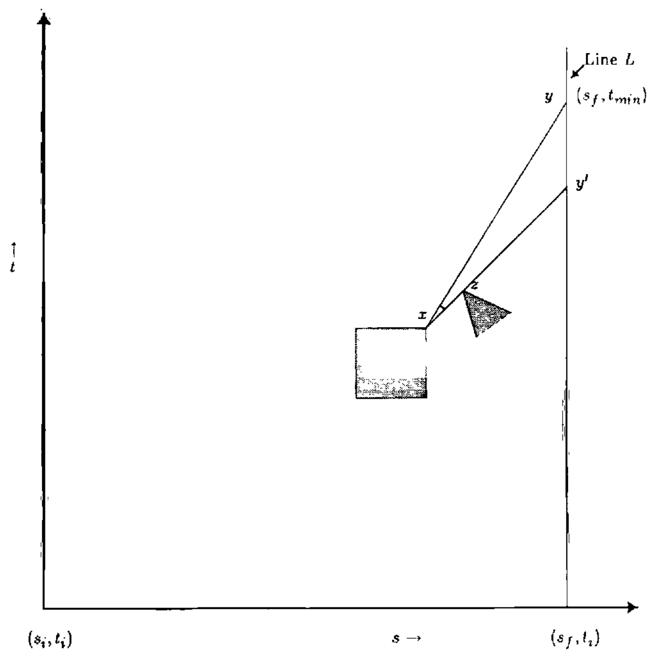


Fig. A.4. Illustration for the proof of the MinTime-node Lemma in s × t *space with slope constraint.*

point x' from which y is cone-visible, by a *shorter* path φ composed of straight line segments with slope $\leq V_{max}$, joining the vertices of obstacles.

$\Rightarrow$ The point $y \notin$ an optimal path from x' to z.

II.C. MinTime-node Lemma in $s \times t$ Space with Slope Constraint

The minimum time node (s_f, t_{min}) is an intersection point of line L and a line segment of slope V_{max} (or $-V_{max}$) emanating from an obstacle vertex.

The MinTime-node Lemma follows as a corollary from the above proof. The aim is to reach the final position s_f in minimum time. Instead of one goal node, a set of goal nodes is given. This set is the line L drawn from (s_f, t_i) parallel to the t axis, as shown in Fig. A.4. The minimum time goal node has to lie on L. What remains to be proven is that it also lies on a line segment with slope V_{max} (or $-V_{max}$) emanating from an obstacle vertex. We prove it for the positive case. The proof for the negative case is similar.

Let t_{min} be the minimum time that the position s_f may be reached in. Now consider the problem of finding a collision-free path from initial node $I = (s_i, t_i)$ to final node $F = (s_f, t_{min})$.

By the MinPath Theorem in $s \times t$ space with slope constraint, the minimum length path to node F is composed of straight line segments (*slope* $\leq V_{max}$) joining vertices of the obstacles. Let the last segment in the path be $\overline{xy}$, where point x corresponds to a vertex of an obstacle and y corresponds to node F. We claim that slope $(\overline{xy}) = V_{max}$.

To prove this, assume otherwise. There are two possibilities:

1. $slope(\overline{xy}) > V_{max}$, or
2. $slope(\overline{xy}) < V_{max}$.

The first possibility is ruled out by the slope constraint. For the second, assume that $slope(\overline{xy}) < V_{max}$. Sweep the ray $\overline{xy}$ with pivot at x to increase the slope till it reaches the value V_{max} or the ray hits an obstacle vertex like z (see Fig. A.4). If the slope does reach the value V_{max}, it implies that $\exists$ a path with t coordinate $< t_{min}$. If the ray hits an obstacle before this happens, it implies that $\exists$ a path with t coordinate $\leq t_{min}$. This path comprises two line segments $\overline{xz}$ and $\overline{zy'}$ where y' is the new intersection of the swept ray with line L. In either case, there is a contradiction. It follows that $slope(\overline{xy}) = V_{max}$. Hence, the minimum time node lies on a line segment with slope V_{max} emanating from an obstacle vertex.

References

Brady, J. M., et al. 1983. *Robot motion: planning and control.* Cambridge: MIT Press.

Brooks, R. A. 1982 (May). Solving the find path problem by representing free space as generalized cones. AI Memo 674. Cambridge: Massachusetts Institute of Technology.

Brooks, R. A., and Lozano-Perez, T. 1983 (Karlsruhe, West Germany). A subdivision algorithm in configuration space for findpath with rotation. *Proc. 8th Int. Joint Conf. Artificial Intell.*, pp. 799–806.

De Boor, C. 1978. *A practical guide to splines.* New York: Springer-Verlag.

Donald, B. R. 1983 (June). Hypothesizing channels through free space in solving the find-path problem. AI Memo 736. Cambridge: Massachusetts Institute of Technology.

Hopcroft, J., Joseph, D., and Whitesides, S. 1984. Movement problems for 2-dimensional linkages. *SIAM J. Computing* 13(3):610–629.

Kant, K., and Zucker, S. W. 1984 (Montreal). Trajectory planning problems I: determining velocity along a fixed path. *Proc. IEEE 7th Int. Conf. Pattern Recognition,* pp. 196–198.

Kant, K., and Zucker, S. W. (1986, June, Toulouse, France). Planning smooth collision-free trajectories: path, velocity and splines in free-space. *8th IASTED Int. Symp. Robotics and Artificial Intell.*

Lozano-Perez, T. 1980 (Dec.). Spatial planning: a configuration space approach. AI Memo 605. Cambridge: Massachusetts Institute of Technology.

Lozano-Perez, T., and Wesley, M. 1979. An algorithm for planning collision-free paths among polyhedral obstacles. *Comm. Assn. Computing Machinery.* 22(10):560–570.

Morovac, H. P. 1980. Obstacle avoidance and navigation in the real world by a seeing robot rover. Ph. D. thesis. Stanford University, Department of Computer Science.

O'Dunlaing, C., and Yap, Y. K. 1983 (March). The Voronoi method for motion-planning: the case of a disc. TR No. 53. New York, Courant Institute of Mathematical Sciences, Department of Computer Science.

Paul, R. P. 1981. *Robot manipulators: mathematics, programming and control.* Cambridge: MIT Press.

Reif, J. H. 1979. Complexity of the mover's problem and generalizations. *Proc. IEEE 20th Symp. Foundations Computer Science,* pp. 421–427.

Reif, J. H., and Sharir, M. 1985 (April). Motion planning in the presence of moving obstacles. TR-06-85. Cambridge: Harvard University, Center for Research in Computing Technology.

Schwartz, J. T., and Sharir, M. 1982*a* (Feb.). On the piano movers' problem: II. General techniques for computing topological properties of real algebraic manifolds. TR No. 41. New York, Courant Institute of Mathematical Sciences, Department of Computer Science.

Schwartz, J. T., and Sharir, M. 1982*b* (Sept.). On the piano movers' problem: III. Coordinating the motion of several independent bodies: the special case of circular bodies moving amidst polygonal barriers. TR No. 52. New York, Courant Institute of Mathematical Sciences, Department of Computer Science.

Udupa, S. 1977. Collision detection and avoidance in computer controlled manipulators. Ph. D. thesis, California Institute of Technology.

Wallace, R. 1984. (Austin, Texas). Three findpath problems. *Proc. AAAI Nat. Conf. Artificial Intell.*

Wangdahl, G. E., Pollock, S. M., and Woodward, J. B. 1974. Minimum-trajectory pipe routing. *J. Ship Res.* 18(1):46–49.