Accessibility: A New Approach to Path Planning among Moving Obstacles*

Kikuo Fujimura
Hanan Samet

Center for Automation Research and
Computer Science Department and
Institute for Advanced Computer Studies
University of Maryland
College Park, Maryland 20742 USA

ABSTRACT

A study is performed of the problem of planning a collision-free path for a robot in a time-varying environment. It is assumed that an obstacle moves along a known path. Our formulation also allows the destination point (the point to be reached) to move along a pre-determined path. A new concept of 'accessibility' from a point to a moving object is introduced and is used to determine a collision-free path. An environment which contains some uncertainty in which the robot needs to see and possibly plan an alternative path is also considered. The ability to deal with moving obstacles is useful in a variety of visual tasks such as the navigation of an autonomous vehicle and the interaction between robot arms.

Keywords: motion planning, moving obstacles, uncertainty

1. Introduction

Motion planning is a problem of fundamental importance in visual navigation as well as in robot arm manipulation [6, 7, 10, 11]. The problem is to find a path that connects a given start and final configuration, among a pre-defined set of obstacles, so that the path is collision-free. For the sake of simplicity, the obstacles are usually approximated by polygons, and the object that follows the path (usually termed a *robot*) is treated as a point.

In this paper, we consider the two-dimensional path planning problem in which the obstacles, as well as the destination point, are in motion. The ability to perform path planning among moving obstacles and a moving destination point can be useful to various robot systems equipped with vision. For example, the ability to avoid moving obstacles leads to an increase in the mobility of a robot for navigation. It also results in a higher productivity for factory manipulators. By allowing the destination point to move, we are able to consider a larger class of robot applications. For example, suppose that the robot arm must pick up an object that lies on a conveyer belt with visual guidance. As another example, consider an autonomous vehicle whose goal is to catch up with another vehicle that is in motion by tracking its movement. We assume that all the motions of obstacles are piecewise linear - i.e., the motion of an obstacle consists of a sequence of intervals where in each interval the obstacle moves in a fixed direction at a constant speed without rotation.

The piecewise linear motion assumption simplifies the problem, as well as makes the task of the visual component easier. First, we assume that all the motions of the obstacles have already been measured prior to the planning stage. Later, we discuss the avoidance of obstacles with an uncertain velocity where the robot has to alternate between the actions of seeing and moving.

The dynamic nature of the problem gives rise to some new aspects in our problem that are not considered in the path planning problem among stationary obstacles. For example, a path that is viable at a particular instance of time may no longer be executable at another instance of time. In addition, there are two types of optimal paths, i.e., the one that reaches the destination by traversing the shortest distance, and the one that reaches the destination in the shortest time. These two optimal paths are usually not identical.

Reif and Sharir [9] show that motion planning for a three-dimensional environment containing moving obstacles is PSPACE-hard given bounds on the robot's velocity, and NP-hard without such bounds. Canny and Reif [2] show that motion planning for a point in the plane with a bounded velocity is NP-hard, even when the moving obstacles are convex polygons moving at constant linear velocity without rotation. Nevertheless, there have been some approaches at solving the problem. These methods are successful in a limited domain. Kant and Zucker [5] decompose the problem into two parts. In the first part, they ignore the moving obstacles in planning a path among the stationary obstacles. In the second part, a graph is used to define regions through which the robot may not pass when following the path computed in the first part. The positions of these regions influence the choice of the velocity. Erdmann and Lozano-Perez [3] make use of stacks (i.e., piles) of two-dimensional Configuration Spaces. These spaces are created each time some object changes its velocity. A path consists of a sequence of vertex-to-vertex transitions between two adjacent elements of the piles. None of these approaches deal with the case in which the destination point can also be in motion.

Our approach is as follows. We assume that the robot can move faster than the obstacles. We first determine the set of points called a collision front with respect to the starting point. These are the points where our point robot meets with an obstacle for the first time when the robot is moving at a constant velocity. In general, the front is in the form of a curve. We move the robot to one of the end points, say E, of the collision fronts. From E, we again determine the collision fronts with respect to E, and move the robot from E to one of the end points of the new collision fronts. We repeat this

* The support of the National Science Foundation under Grant DCR-86-05557 is gratefully appreciated.

process until the destination point is finally met.

The rest of this paper is organized as follows. Section 2 introduces the concept of accessibility. Section 3 presents the algorithm to find a collision-free path using accessibility, and Section 4 analyzes the execution time of the algorithm. Section 5 extends the concept of accessibility to piecewise linear motion of the obstacle, and applies the concept in an environment with uncertainty. Section 6 compares our approach with some other approaches. Section 7 contains concluding remarks.

2. Accessibility

Throughout the paper, G and R are used to denote the destination point and the point robot, respectively. In this section, in order to ease the presentation, the obstacles are assumed to make a straight motion. This assumption is extended to allow piecewise linear motion in Section 5.

First, we define our obstacles. An obstacle is a convex polygon which moves in a constant direction at a constant speed. We call such a straight motion a *movement*. For the sake of explanation, we treat the line-segments (edges) that constitute polygons as the basic units of our discussion. A *movement* is defined by a tuple (L, d_L, v_L) which represents the motion of line-segment L in direction d_L at speed v_L. We will consider an environment which contains a finite set of movements, $M=\{M_1, M_2, \ldots, M_n\}$, where each M_i represents a movement as defined above. Note that M corresponds to the motions of all the edges in the environment; not just one polygon. Hence, if a polygon P_i consists of l_i edges, then $n = l_1 + l_2 + \cdots + l_k$, where k is the number of polygons in the environment.

Second, we describe the motion of R, the point robot. After leaving the start point at a start time, R can take any motion as long as its speed doesn't exceed a given maximum speed. We assume that the maximum speed of R is greater than that of any of the obstacles and that of the destination point.

Third, the path of G is piecewise linear; i.e., it consists of a finite series of movements. Within a movement, G moves in a constant direction at a constant speed.

Consider a set of movements $M=\{M_1, M_2, \ldots, M_n\}$ and G, the destination point. Let R be a point robot located initially at O at time t_0. Suppose that R starts moving at time t_0 at a speed v in a fixed direction. A point V (V is either G or a vertex of a polygonal obstacle) is said to be *accessible* from O, if there exists a direction of the motion of R such that R meets V without a prior interception by any other movements. We say that V and R *meet* if there exists a location X through which both V and R pass at the same time t, where $t_0 < t$. The location X is called an *accessible point* of V, and t is called the *accessible time* of X.

The accessible point varies for different values of the speed and the initial location of R. Therefore, we use $V(O, t_0, v)$ to denote the accessible point of V formed by R whose initial location, start time and speed are O, t_0, and v, respectively. Also if V is stationary, then the accessible point of V is V itself, if applicable.

Let VS be the set of vertices of the given polygonal obstacles in motion M (termed a *vertex set*), and let O, t_0, v be as in the previous definition. The set of accessible points of vertices in VS with respect to O, t_0, v are called the *accessible point set* and denoted as $APS(M, O, t_0, v)$. Since some vertices in VS may not be accessible from O, the size of APS is at most $|VS|$.

3. Motion Planning Algorithm

This section describes our algorithm for path planning using the accessibility concept introduced in Section 2. Let $V_a V_b$ be an edge of an obstacle and let P_a and P_b be accessible points of V_a and V_b, with respect to R's initial location O, start time t_0, and speed v. If R keeps moving in a direction between OP_a and OP_b at speed v, then R will eventually collide with edge $V_a V_b$ at some point. The set of these collision points with respect to edge $V_a V_b$ forms a (curve) segment, which is called the *collision front* of edge $V_a V_b$ (Figure 1). On the other hand, when R keeps moving outside the angle formed by OP_a and OP_b at speed v, then R does not meet $V_a V_b$ in motion. This fact is used to guide R in reaching the destination point. R can move from O to one of the accessible points of $APS(M, O, t, v)$, say Q, without colliding with any obstacle. At Q, we construct another APS with Q as the initial point. Then we can again move to a newly generated accessible vertex, say Q', without colliding with any other movement. The motivation behind our approach to path planning is to repeat this process until R eventually reaches the destination point.

Now, we present the procedure to find a path to the destination point using APS. Let S be a start point (i.e., the point at which R, the point robot, is initially found), t_0 be a start time, G be the destination point which is in motion, and $M=\{M_1, \ldots, M_n\}$ be the set of movements. With each accessible point, say P, we also associate a time value,. say $t(P)$, to denote P's corresponding accessible time. We use a priority queue of points where the accessible time of each point serves as the point's priority.

procedure *FINDPATH*

```
(1)  Insert S in the queue and set its default accessible time
     to $t_0$.
(2)  Pop a point, say P, from the queue whose associated
     minimum accessible time is the minimum (i.e., it is the
     youngest).
(3)  If P in step (2) is the destination point, then exit the
     procedure; otherwise, construct $APS(M,P,t(P),v)$ and
     enter the new points of APS into the queue. Repeat
     steps (2) and (3).
```

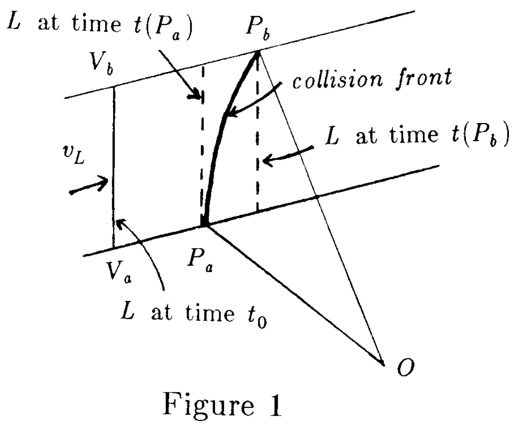

Figure 1

Figure 2 is an example of the result of planning a path using our approach. S and G are the start and destination points. Triangle ABC is an obstacle moving towards the right, and rectangle $DEFG$ is an obstacle moving towards the left. Points N, P, and Q are the accessible points of B from S, of F from N, and of H from P, respectively. The dashed objects show the locations of the corresponding obstacles at the indicated times. Note that while the robot is moving between P and Q, it is coincident with some point on edge FH of obstacle $DEFH$. $SNPQG$ constitutes the final path.

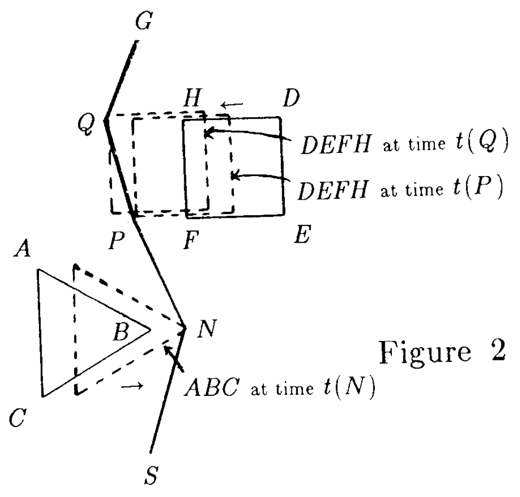

Figure 2

4. Execution Time

This section analyzes the time required to generate a collision-free path using the algorithm described in the previous section. First, we show that the shape of the collision front is either a parabola, hyperbola or an ellipse. Without a loss of generality, consider the situation shown in Figure 3. Line segment L lies parallel to the y-axis. Let (x_0,y_0) be the location of one of L's endpoints at the start time, l be the length of L, v_L be the velocity of L, v be the velocity of the robot, and θ be the angle formed by the direction of the movement and the x-axis. Suppose that a point P at (x_0,Y) in L is accessible from the origin (0,0) at point (x,y). x and y must satisfy:

$$t(x,y)=\frac{\sqrt{x^2+y^2}}{v_{\max}}=\frac{\sqrt{(x-x_0)^2+(y-Y)^2}}{v} \tag{1}$$

$$y=(x-x_0)\tan\theta+Y \tag{2}$$

$$y_0\leq Y\leq y_0+l \tag{3}$$

Equations (1) and (2) define a quadratic relationship between x and y - i.e., the collision front lies either on a parabola, hyperbola, or ellipse. (The curve degenerates to a straight line when the direction of the motion is parallel to L). Also, we can show that the origin is on the same side as one of the foci of the curve. Generally, the two endpoints of a collision front

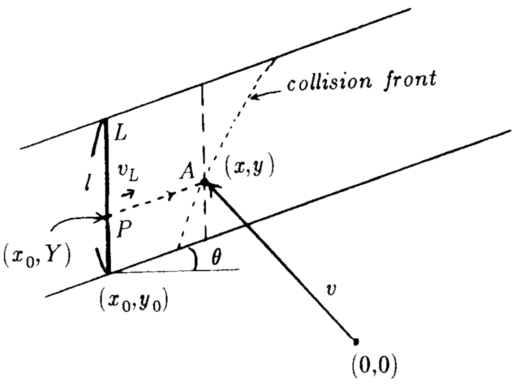

Figure 3

(accessible points) can be computed by setting Y to y_0 and y_0+l in equations (1) and (2).

Having computed two accessible points of a single movement, we must now compute all the accessible points (APS) when the robot is at a given start point, say S, with a given speed v. For a set of polygonal obstacles with n vertices, there are n candidate accessible points. However, some points may not be accessible from S since some other movement intercepts the accessibility of that point. We now show that it takes $O(n\log n)$ time to compute APS. The set of accessible points is determined by using a similar technique to that used to compute the visibility of a given set of line segments. This is also known as a plane-sweep algorithm [8]. This is a two-step process. The first step sorts all the vertices, say in a clockwise direction with respect to S. This sorting process takes $O(n\log n)$ time. The second pass rotates a line about S and halts each time the line intersects a vertex, and we check whether or not the collision front associated with the vertex is accessible from S. This process can be achieved in $O(\log n)$ time by using a 2-3 tree [1] to maintain the active collision fronts based on their distance from S. In this way, the closest collision front is marked as accessible from S. Since it takes at most $O(n\log n)$ time to build the initial 2-3 tree and $O(\log n)$ time for each update at a vertex, the determination of accessibility of n candidate points takes $O(n\log n)$ time. Therefore, given a start point and a set of candidate points, it takes $O(n\log n)$ time to compute accessible points from the start point.

As there are n vertices in the environment, it can be shown [4] that an APS needs to be generated at most n times before a path reaches the destination point This means that procedure *FINDPATH* require $O(n^2\log n)$ time to compute a path. Note that for stationary polygonal obstacles, using a visibility graph, the shortest length path can be computed in $O(n^2)$ time [12].

5. Two Adaptations

In this section, we describe two adaptations of the accessibility concept in an environment in which (1) the motion of the obstacles is piecewise linear, and (2) there is some uncertainty in the motion of the obstacles.

5.1. Piecewise linear motion of the obstacles

At times, a straight motion is not sufficient to describe the movement of the obstacles. A somewhat more general description is a piecewise linear motion- i.e., the motion consists of a finite number of time intervals such that during each interval the obstacle moves in a fixed direction with a constant speed. It is simple to extend the accessibility concept to this situation. Now, we can define a movement for line-segment L as (L, d_L, v_L, TI_L), where TI_L represents the time interval during which L moves in direction d_L at speed v_L. When the motion of the obstacle is piecewise linear, then the collision front becomes a finite set of connected sub-segments of quadratic curves. Note that points where two curves are connected may now correspond to an internal point of the line segment in motion- i.e., L.

The value n in the execution time analysis (Section 4) now represents the total number of movements made by the polygonal obstacles. In other words, if a polygon P_i consists of l_i vertices and each polygon P_i has m_i movements in its life time, then $n = l_1 m_1 + l_2 m_2 + \cdots l_k m_k$, where k is the total number of polygons.

5.2. Environment with uncertainty

It should be pointed out that our approach can be easily adapted to an environment with uncertainty. Such a situation arises due to errors in the measurement of the motion of the objects. For example, the notion of accessibility can be extended to handle some amount of uncertainty in the speed of the obstacles. Consider a line segment L which moves in a constant direction at a speed v, where $v_1 \le v \le v_2$ for some known v_1 and v_2. We can create two collision fronts, C_1 and C_2, which correspond to the movement of L at speeds v_1 and v_2, respectively. A collision front of L at velocity v must lie between C_1 and C_2. We use the term *collision area* to denote the area which is swept by L and bounded by $C1$ and $C2$ (Figure 4). The two extremum points of this area (e.g., points A and B in Figure 4) can be used as accessible points in procedure *FINDPATH* (described in Section 3) to find a path among obstacles with uncertain speeds. At an accessible point, the robot may have to 'see' again to make sure that it knows the location of the other obstacles. After 'seeing', the robot moves toward the next accessible point. In other words, the robot need not 'see' during the time interval.

Another source of uncertainty is the point robot itself. Due to errors inherent to the driving mechanism or minor turbulence along the path, the robot may have a certain range of uncertainty with respect to its own velocity. A dynamic nature of the problem aggravates the problem: once a robot falls behind a strictly planned path specified with time, it becomes increasingly hard or even impossible for the robot to catch up with the original path. This is because a small delay at an earlier stage of the path can easily grow to be an insurmountable delay from the pre-planned path, as the delay increases with its propagation. To prevent this from happening, we can incorporate the uncertainty factor of the robot's velocity by defining a collision area similar to that shown above. In the future, we will also take into account uncertainty in the direction of the velocity.

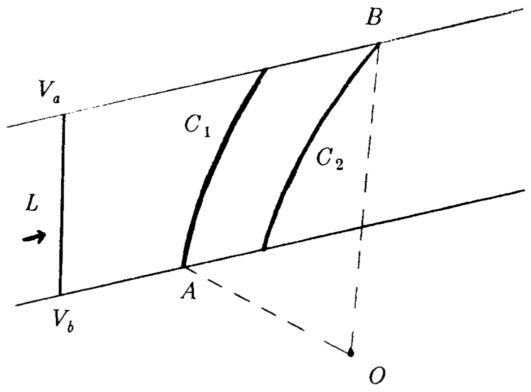

Figure 4

6. Discussion

At this point, let us compare our approach with some past work in the field of motion planning among moving obstacles. In particular, we pay attention to how 'fast' the final path is. Our approach is in contrast with the ones taken by Kant and Zucker [5] and Erdmann and Lozano-Perez [3]. Lee and Lee [6] use an approach that is similar to [5].

Kant and Zucker decompose the process of path planning into two steps, i.e., "where" and "when" phases. In the first step, they determine "where to go", i.e., they plan a path among stationary obstacles. Next, in the second step, the speed along the fixed path, which is an output of the first stage, is varied so that the trajectory is collision-free with the moving obstacles. However, their algorithm become inefficient, and may even fail to generate a path, when one of the obstacles moves on a path which is coincident with the path of the robot which was computed in the first stage of the algorithm. The problem is that the path is fixed in the second stage, and thus the robot is not allowed to circumnavigate moving obstacles. Also, as the path is fixed in the first stage, their method cannot easily incorporate a moving destination point.

Erdmann and Lozano-Perez [3] represent the movements of the obstacles as a set of slices which embodies space-time. These slices represent the Configuration Spaces at the particular times. The times are those at which some moving obstacle changes its velocity. A path consists of a set of path segments which starts at a vertex of an obstacle in one slice and terminates at a vertex of an obstacle in the next slice. Between two vertices, the moving obstacle makes a straight movement with a constant speed. As a result, along a final path, a point-robot changes its velocity only at some of the vertices of the obstacles when some obstacle's velocity changes. Their approach is complete when the topology of the free space doesn't change, (i.e., the obstacles don't merge or split) and runs in time $O(rn^3)$, where n is the total number of edges in the environment and r is the number of slices constructed. Their path can be inefficient when obstacles don't change their velocities at all, since the robot is forced to move at a constant speed from start to end. It is possible to incorporate the case of a moving destination in the space-time approach.

Our approach determines "where" and "when" to go at the same time. As a result, our approach generates a collision-free path also in a situation where the approach of

Kant and Zucker fails. A path generated in our method also consists of straight path segments, but they do not necessarily terminate at a vertex of an obstacle. Our approach is complete only when the topology of free space doesn't change. For example, we can imagine a 'gate' on the route to the destination which opens only for a specific short period of time. Then, the planner must be able to generate a path such that the robot arrive at the gate just in time for the opening. In particular, the planner must be at least able to generate a pointwise time-optimal path.

Figure 5 shows how the above three approaches negotiate a moving obstacle. S and G represent the start and destination points. Object $ABCD$ is an obstacle of size 2m by 2m whose initial speed is 2 m/s moving towards the right. $ABCD$ changes its direction of motion towards the left when its right edge reaches point X and also slows down its moving speed to 1 m/s. Three paths which are the outputs of the above mentioned approaches are illustrated. Here, we impose a speed limit 3 m/s on the robot. Path α is the one planned by our algorithm. Path β is generated by vertex-vertex transitions as implemented by Erdmann and Lozano-Perez. The first vertex-vertex transition can be either S to C at Z or S to D at W. The later possibility must be eliminated in this situation because of the speed bound. Path γ is a path that would be generated using Kant and Zucker's approach. It is a straight line, since there are no stationary obstacles in the scene. The second stage determines the speed along this straight line. Note that along path γ the robot gains speed after passing vertex C while moving towards the left, and again after passing vertex D. Figure 6 is the time-speed diagram for these paths. Paths β and γ terminate at the destination later than path α. This indicates that paths generated by their corresponding algorithm are, in general, not time-optimal.

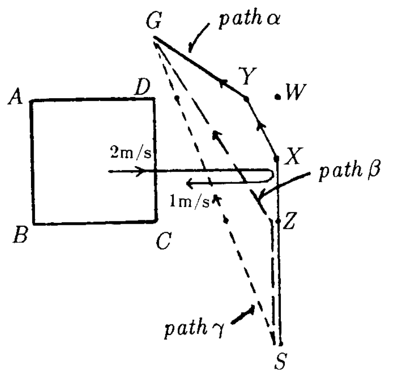

Figure 5

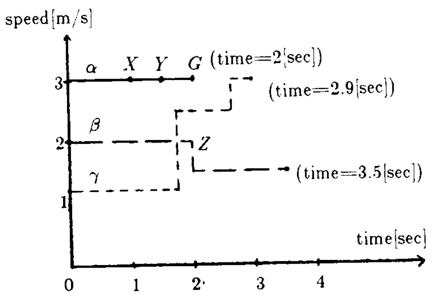

Figure 6

7. Conclusions

We have presented a new approach to path planning among moving obstacles using the concept of accessibility. In our formulation, the destination point was also permitted to move, which was not allowed in any of the previous approaches. In so far as described above, the concept of accessibility can be effectively utilized in planning a path among obstacles whose motions are piecewise linear. We are currently studying the time optimal property of the algorithm.

References

1. A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *The Design and Analysis of Computer Algorithms,* Addison-Wesley, Reading, MA, 1974.
2. J. Canny and J. Reif, New lower bound techniques for robot motion planning problems, *Proceedings of the 27th IEEE Symposium on Foundations of Computer Science,* Los Angeles, CA, October 1987, 49-60.
3. M. Erdmann and T. Lozano-Perez, On multiple moving objects, *Algorithmica 2,* 4(1987), 477-522.
4. K. Fujimura and H. Samet, Time-minimum paths among moving obstacles, in preparation, 1988.
5. K. Kant and S.W. Zucker, Toward efficient planning: the path-velocity decomposition, *International Journal of Robotics Research 5,* 3(Fall 1986), 72-89.
6. B. H. Lee and C. S. G. Lee, Collision-free motion planning of two robots, *IEEE Transactions of Systems, Man, Cybernetics 17,* 1(January/February 1987), 21-32.
7. T. Lozano-Perez and M.A. Wesley, An algorithm for planning collision free paths among polyhedral obstacles, *Communications of the ACM 22,* 10(October 1979), 560-570.
8. F. P. Preparata and M. I. Shamos, *Computational Geometry: An Introduction,* Springer-Verlag, New York, 1985.
9. J. Reif and M. Sharir, Motion planning in the presence of moving obstacles, *Proceedings of the 25th IEEE Symposium on Foundations of Computer Science,* Portland, OR, October 1985, 144-154.
10. J. T. Schwartz and M. Sharir, On the piano movers' problem: III. Coordinating the motion of several independent bodies: The special case of circular bodies moving amidst polygonal barriers, *International Journal of Robotics Research 2,* 3(Fall 1983), 46-75.
11. S. H. Whitesides, Computational geometry and motion planning, in *Computational Geometry,* G. T. Toussaint, Ed., North-Holland, Amsterdam, 1985, 377-427.
12. E. Welzl, Constructing the visibility graph for n line segments in $O(n^2)$ time, *Information Processing Letters 20,* 4(May 1985), 167-171.