Velocity planning for a robot moving along the shortest straight line path among moving obstacles *

K. Krithivasan † A.Rema Stefan Schirra ‡ P.I. Vijaykumar

Abstract

In the earlier studies on collision-free motion planning the main aim has been to minimize the time taken by the moving robot to reach the goal. In this paper we discuss the problem of planning the velocity for a robot along the shortest distance path between two points in space, avoiding collision with the moving obstacles. By using a space-time transformation we show that it is possible to determine a collision-free motion along the straight line joining the source point to the goal point in $O(n \log n)$ time, optimizing on the time taken for the shortest length path.

Introduction

The problem of motion planning is to determine the existence of a path for a moving object from a given source point to a desired destination point, avoiding collision with any obstacles in the environment. The static domain, where the obstacles are stationary, has been studied extensively in [12], [6] and [4]. More specifically, the problem of computing the Euclidean shortest length path in the plane amidst stationary obstacles has been studied in [8], [10], [5] and [9]. Tools such as the visibility graph and the shortest path map have been used to map the connectivity of free space and the shortest path is obtained by searching these data structures. In the static domain the shortest path will necessarily coincide with the time-minimal path, assuming the moving obstacle is moving at all stages with its maximum velocity modulus. However in the dynamic domain, when we consider motion planning in the presence of moving obstacles, the shortest length motion may not necessarily coincide with the time-minimal motion.

Most earlier studies on motion planning in dynamic domains have considered time minimal paths from the source to the goal as in [7] and [1]. Here the prime consideration has been to minimize the time taken to reach the goal, and hence during most or all sections of its path, the robot Z moves at its maximum speed v_{max}. However in certain situations it will be more relevant to consider paths of minimum distance, rather than paths of minimum time, especially when the cost per unit length of distance traveled is high, or when it is just sufficient that Z reaches the goal. In this paper we consider the case of finding the shortest length path from a start point S to a destination point G, in the presence of moving obstacles in the environment. The algorithm has been developed for obstacles moving in the plane.

When we consider the case of finding the Euclidean shortest length path in the presence of moving obstacles, the problem becomes simple, since the shortest length path is trivially along the line joining the source and the goal. Note however that this is not necessarily a time minimal path. The algorithm has to mainly ensure that the path defined does not collide with any of the moving obstacles as it travels from the start to the destination. Since the prime criterion is to minimize the distance traveled, it is possible that along its path the object may travel with speed less than its maximum speed

*This work was partially supported by the Indo-German project on Computational Geometry and its applications, sponsored partially by MHRD, India and KFA, Germany and the DST project on Computational Geometry.

†Dept. Of Computer Science and Engineering, Indian Institute Of Technology, Madras, India.

‡Max-Planck-Institut für Informatik, Saarbrücken, Germany.

v_{max} in some sections to avoid collision, and may even be stationary for some finite amount of time.

Terms and definitions

The moving object or robot will be denoted by Z. Z is a point object. The problem of finding the Euclidean shortest distance path in the plane, in the presence of moving obstacles may be stated as:

Given an environment containing polygonal obstacles in translational motion, where n is the total number of vertices in the scene, with Z initially at S (the start point), and v_{max} being the maximum speed of Z, determine the speed profile for Z along SG such that the time taken to reach G (the goal point) is minimized, and such that at all stages Z travels with a speed less than or equal to v_{max}, avoiding collision with any of the moving obstacles; else report that a motion cannot be found.

We have approached the problem using the space-time transformation (described subsequently) and the plane sweep method. The obstacles are all simple polygons, each having a velocity modulus greater than zero (we first deal with the case of constant velocity and then consider acceleration). S is the start point and G the goal point. We may assume that SG lies along the positive x-axis with S at $(0,0)$. We shall use SG to refer to the open line segment joining S and G. Our algorithm is based on the assumption that all the obstacles are moving with nonzero velocity modulus and that there is no collision between the obstacles.

Frame of reference

The shortest length path is along the line joining S and G, which in our case is along the x-axis, starting at the origin. The path of Z has to be planned avoiding collision with the obstacles as they cross the x-axis. One way of capturing the motion of Z amidst the obstacles is to map the positions of the obstacles as they cross the x-axis and the time at which they cross the x-axis, to an $x-t$ frame of reference. We shall consider only those obstacles which intersect the x-axis at some point of time in their

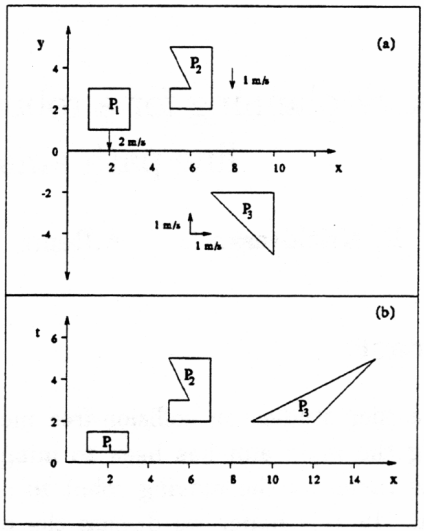


Fig-1: (a) Obstacles in the xy-plane
(b) Obstacles in the xt-frame

path. For each such obstacle, we can compute the time at which each vertex crosses the x-axis and also the position at which it crosses this axis. Given the initial position of each vertex, and the speed of each obstacle with its direction of motion, this computation can be done in constant time for each vertex. The position and time are then mapped onto the x-t frame, with the position along the x-axis, and the time along the y-axis.

Now, in this new frame of reference we have a snapshot of each obstacle as it crosses the x-axis. Since we wish to find a collision-free path for Z from S to G (where S and G are on the x-axis) and since collisions may occur precisely at the time when the obstacles cross the x-axis, in this new frame of reference we may consider each obstacle to be stationary for the time under consideration and plan a collision free path for Z accordingly. For example, Fig-1(a) shows a scene with three moving obstacles and Fig-1(b) shows the same obstacles transformed to the $x-t$ frame of reference. For the case of constant velocity, the obstacle edges map on to straight line segments. This will not be the case for acceleration, however we consider only constant velocity in the following discussion.

Since no two obstacles collide in the $x-y$ frame of reference, no two obstacles overlap in the $x-t$ frame of reference, except perhaps at the edges. We are interested in finding a path for Z along SG from $x=0$ to $x=G$, avoid-

ing collision with any obstacle. The position and instant when such a collision is possible with a particular point on or in the interior of a polygon is given by the ordinate and abscissa of that point in the $x-t$ plane. Now, in the new frame of reference, our problem may be restated as:

Determine a path for an object Z, moving amidst stationary obstacles (in the $x-t$ frame) from a point S on the line $x=0$ at time $t=0$, to a point on the line $x=G$, minimizing the time taken, i.e. meeting the line $x=G$ at the lowest possible point.

Properties of the path of Z in the $x-t$ frame

Let v_{max} be the maximum speed attainable by Z and let $m = 1/v_{max}$. In the $x-t$ plane, Z is initially at the point $(0,0)$ and has to reach some point on the line $x=G$, avoiding collision with the interior of any of the obstacles (which may be considered stationary in the new frame of reference.) We shall now consider the properties of the trajectory of Z in this new frame of reference. (The word trajectory is used here since we are referring to the speed of Z along its path as a function of time.)

1. Assuming that at each stage Z moves with uniform speed, changing its speed only when it encounters an edge or a vertex, the path of Z will be the union of piecewise linear line segments, which do not intersect the interior of any obstacle.

2. No line segment constituting a path of Z will have slope less than m, since a slope less than m implies Z moves at a speed which is greater than v_{max}, which is not possible. A section of the trajectory parallel to the y-axis implies Z is stationary at this point.

Constant Velocity

A **visibility edge** is defined to be a straight line segment of positive slope $\geq m$, joining two vertices in the planar $x-t$ graph G, such that it does not intersect the interior of any obstacle.

A **path** of Z is defined to be the union of piecewise linear segments each of which have positive slope $\geq m$ and do not intersect the interior of any obstacle. It can be seen that a speed profile (path in the $x-t$ frame) for Z can be obtained as a union of obstacle edges and visibility edges (each of slope $\geq m$) by the visibility graph method which takes $O(n^2)$, where n is the number of vertices in the $x-t$ frame.

Two paths are said to be **equivalent** if they start and terminate at the same point.

A *path* is said to be an **α-path** if it is the union of piecewise linear line segments which do not intersect the interior of any obstacle and are

1. either vertical or
2. have a slope $= m = \frac{1}{v_{max}}$ or
3. part of an edge of an obstacle whose slope is $\geq m$.

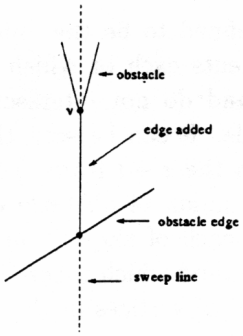
Lemma There exists an equivalent α-path for every *visibility edge* that is not an obstacle edge of the straight line planar graph G in the $x-t$ frame.

The proof of the above lemma has been omitted. The lemma ensures that for every path of Z in the $x-t$ frame consisting of visibility edges and obstacle edges of slope $\geq m$, an equivalent α-path can be obtained. It suffices thus to look for an α-path that intersects the line $x=G$ at the lowest possible point. We find such a path by a plane sweep technique.

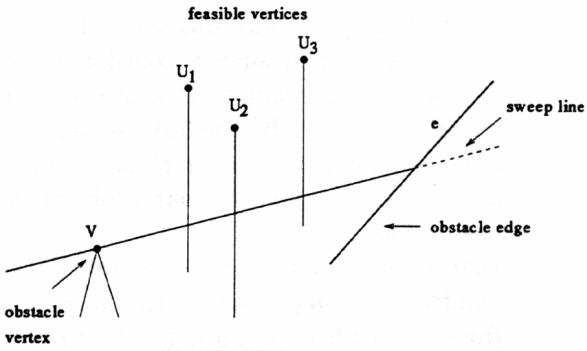
Algorithm

1. Perform a horizontal plane sweep of the obstacles in $x-t$ space from left to right. The sweep line is vertical. The sweep line status is a sorted list of edges intersecting the sweep line. This list can be maintained with $O(\log n)$ operations at each vertex. The sweep line is initially $x=S$ and the line $y=0$ intersects it. The sweep line stops at every vertex.

At every vertex v, add the vertical line segment from v to the obstacle edge intersecting the sweep line just below it, the vertical edge added should not intersect the interior of any obstacle. Maintaining the sweep line status and



HORIZONTAL SWEEP



SLANT SWEEP

finding the obstacle edge just below v can be done in $O(\log n)$ time. Therefore the total time taken for this sweep is $O(n \log n)$.

At each vertex, at most one new vertical line segment and one new point are added to the graph. Therefore $O(n)$ points and edges are added to the graph.

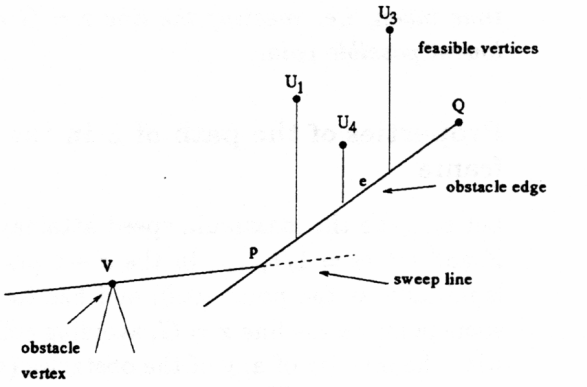
2. Perform a plane sweep with a sweep line of slope $m = \frac{1}{v_{max}}$ in a direction perpendicular to m over the straight line embedded planar graph $x - t$. The sweep line status is a sorted list of the edges intersecting the sweep line. The sweep line initially passes through S and intersects both $x = 0$ and $x = G$. The sweep line stops at all vertices of the $x-t$ graph, i.e. corners of the obstacles and endpoints of the downward vertical line segments added in step 1.

A vertex v is **feasible** if the edge immediately to the left of v intersecting the sweep line, is the line $x = 0$ or the vertex v has been marked *feasible* by a previous vertex.

At each vertex v maintain the sweep line status. If v is a *feasible* vertex,

(a) All the upper end points (vertices) of vertical segments intersecting the sweep line between v and the obstacle edge intersecting the sweep line just to the right of v are marked *feasible*. The vertical line segments are deleted.

(b) If the sweep line intersects an obstacle edge e immediately to the right of v at a point p , all upper end points of vertical line segments which terminate on e above the point p are marked *feasible* and the end point of e above p is also marked *feasible*. The vertical line segments are deleted. Note that the slope of obstacle edge e should be positive and less



SLANT SWEEP

than $\frac{1}{v_{max}}$ otherwise motion along e is not possible and hence no points are marked *feasible*.

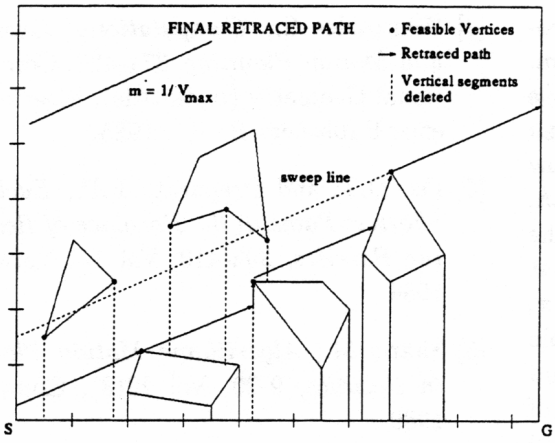
If the edge e is the line $x = G$ stop the sweep and retrace the path of Z.

Complexity

The plane sweep technique takes $O(\log n)$ time at each vertex. Since a vertical segment is deleted whenever a vertex is marked feasible through it, a vertex can be marked feasible only once through a vertical line segment. The total complexity of the algorithm is therefore $O(n \log n)$. In order to retrace a path, whenever a vertex is marked *feasible*, remember the path to the vertex that marked it *feasible*. From the retraced path a speed profile for the robot can be easily calculated.

Constant Acceleration

In this section, we consider the case of constant acceleration. The same algorithm presented for the velocity case is used here but the proof of correctness is different. As before, we perform the transformation to the $x - t$ frame. In



the case of constant velocity, the obstacle edges were mapped on to straight line segments, however in this case, the obstacle edges will be mapped on to parabolic segments of the form $x = \alpha t^2 + \beta t + \gamma$. We need to introduce a few new points as follows,

1. Points on parabolic curves where the slope of the curve is equal to $m = 1/v_{max}$ or 0. This is because the parts of the curve that have slope $< m$ or negative are forbidden for the robot.

2. Consider the initial planar frame of reference. If an obstacle intersects the SG line during its motion, without fully crossing it, at some stage the velocity of the obstacle perpendicular to the SG line must equal 0. (This occurs when the obstacle partially crosses the SG line and starts to reverse its direction). The points on the edges of the obstacle that lie on the SG line at the instant of time when the obstacle reverses its direction (The component of velocity perpendicular to the SG line equals 0) are added to the $x - t$ graph.

It can be seen that the number of new points added are $O(n)$.

Definition : A *path* is said to be an **β-path** if it is the union of piecewise linear or parabolic line segments which do not intersect the interior of any obstacle and are

1. either vertical or

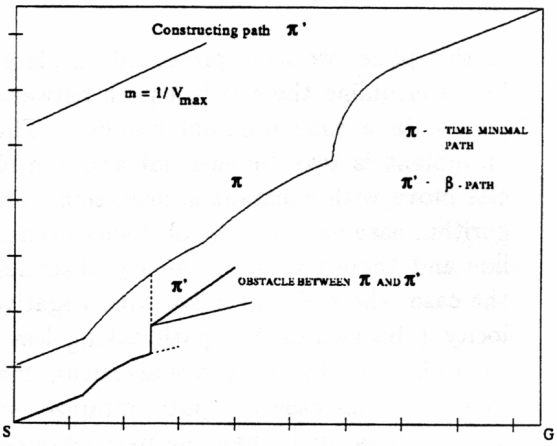
2. have a slope $= m = \frac{1}{v_{max}}$ or

3. part of an edge of an obstacle, in which case they may be parabolic but at no point is the slope of the curve $< m = \frac{1}{v_{max}}$

The same algorithm presented for the constant velocity case is used for the constant acceleration case. We present a short proof here. The following argument refers to the $x - t$ frame, and a path is defined to be any sequence of straight or curved line segments. Let π be a path from $x = S$ to $x = G$ that meets the line $x = G$ at the lowest possible point. π is a path along which the robot Z can move, it has slope greater than or equal to m at all points and it does not intersect the interior of any obstacle. We construct a path π' that has the following properties

1. The path π' always lies below π.

2. There are no obstacles or obstacle edges between π' and π.



The path π' is constructed as follows. Start from the origin $x = S$ and always move along a line of slope $= m$ or along an obstacle edge if an obstacle is encountered. If at any stage, there lies an obstacle between π and π', consider the first such obstacle vertex encountered. We move vertically, on the vertical line segment dropped from this vertex during the horizontal sweep, till the vertex is reached. This ensures that whenever a vertex is encountered between π and π', we move upwards to the vertex such that π' passes through it and hence it will not lie *between* π and π'. Note that when we are moving along a curved path, if we reach a ver-

tex where the slope of the curve $= m$, we proceed on a straight line segment with slope $= m$. This is to ensure that the robot does not move along a curved section of slope $< m$. It can be easily seen that path π' always lies below path π. Since path π' exists and is a $\beta - path$, the algorithm will eventually find path π'. The path π' intersects the line $x = G$ at a point lower than (or at the same point as) the path π. Since π intersects $x = G$ at the lowest possible point, so does π'. Therefore, the path π' found by the algorithm will take the least amount of time to reach $x = G$ from $x = S$. The path π' gives the required speed profile for Z.

Theorem : The Euclidean Shortest distance path amidst obstacles moving with constant acceleration can be found in $O(n \log n)$ time. The path is given as a speed profile of the robot Z along the straight line segment SG joining the start point and the destination.

Conclusion

In this paper we have presented an algorithm for determining the shortest path between two points, in a time minimal manner. The environment is two dimensional and the obstacles move with constant acceleration. The algorithm assumes that the obstacles do not collide and there are no stationary obstacles. In the case where Z can move with negative velocity (backwards) a path taking less time than obtained by the above algorithm may be found. In this case the path obtained will be more time-optimal than the path obtained by our algorithm.

References

[1] Fujimura S., *Motion Planning in Dynamic Domains*, Phd Thesis, University of Maryland, 1989.

[2] Preparata F.P. and Shamos M.I., *Computational Geometry - An Introduction*, Springer - Verlag, 1985.

[3] Latombe J.C, *Robot Motion Planning*, Kluwer Academic Publishers, 1991.

[4] Whitesides S., *Computational Geometry and Motion Planning*, 377-427, Computational Geometry (book title), Elsevier Science Publishers B. V. , 1985.

[5] Lee D.T. and Preparata F.P., *Euclidean Shortest Paths in the Presence of Rectilinear Barriers*, 393-410, Vol 14, Networks, 1984.

[6] Sharir M., *Algorithmic Motion Planning in Robotics*, 9-20, Vol 22(3), Computer, 1989.

[7] Erdmann M. and Lozano-Pérez Tomás, *On Multiple Moving Objects*, 477-521, Vol 2, Algorithmica, 1987.

[8] Hershberger J. and Suri S., *Efficient Computation of Euclidean Shortest Paths in the Plane*, 508-517, Proceedings of the 34th Annual Symposium on the Foundations of Computer Science, 1993.

[9] Mitchell J.S.B, *Shortest Paths among Obstacles in the Plane*, 308-317, Proceedings of the 9th ACM Symposium on Computational Geometry, 1993.

[10] Kapoor S. and Maheshwari S.N., *Efficient Algorithms for Euclidean Shortest Paths and Visibility Problems with Polygonal Obstacles*, 172-182, Proceedings of the 4th Annual ACM Symposium on Computational Geometry, 1988.

[11] Reif J. and Sharir M., *Motion Planning in the Presence of Moving Obstacles*, 144-154, Proceedings of the 26th Annual Symposium on the Foundations of Computer Science, 1985.

[12] Schwartz J.T. and Sharir, M., *A Survey of Motion Planning and related Geometric Algorithms*, 157-169, Vol 37, AI, 1988.