A Maneuvering-Board Approach to Path Planning with Moving Obstacles

Lou Tychonievich David Zaret* John Mantegna
Robert Evans Eric Muehle Scott Martin
Martin Marietta Aero & Naval Systems
103 Chesapeake Park Plaza
Baltimore, Maryland 21220

Abstract

In this paper we describe a new approach to the problem of path planning with moving obstacles. Our approach is based on the maneuvering board method commonly used for nautical navigation. We have extended this method to handle cases where obstacles may maneuver at any time and where knowledge of the position and velocity of the obstacles may be uncertain.

1 Introduction

In this paper, we present an approach to the 2D path planning problem with moving obstacles. We describe a path planner which finds a collision-free path through a field of moving obstacles, from a starting point to a goal point or region, where the goal point or region can itself be in motion. Furthermore, we allow uncertainty in the position and velocity of all objects other than the vehicle for which the path planning is being carried out

We assume that at regular time intervals, or "epochs", the planner receives an updated "scene description", or description of the vehicle's environment; changes in the motion of objects from one epoch to the next can be thought of as representing either unexpected changes in the motion, or corrections to the previous scene description in light of new data. Having received an updated scene description, the planner calculates and returns a vector which represents the course and speed that the vehicle is to follow during the next epoch. Our path planner is, therefore, local and reactive; it determines, at each epoch, a trajectory which is directed as much as possible towards the goal, while at the same time avoids the most immediate obstacles.

Most of the work on path planning that has been reported in the literature has involved a stationary goal point and stationary obstacles (e.g., [Brooks, 1983, Lozano-Perez, 1983, Lozano-Pcrcz and Weley, 1979]. One exception is [Reif and Sharir, 1985], which investigates the computational complexity of motion planning in the presence of moving obstacles for a body in 2D or 3D space. In their paper, Reif and Sharir provide a polynomial time algorithm for what they call the "2D asteroid avoidance" problem: the moving "robot" is a convex polyhedron which moves by translation with bounded velocity modulus, and the obstacles are polygons which move without rotation and with known translational trajectory. Reif and Sharir's approach, however, is essentially "global". That is, their approach involves determining at the outset an entire path from the starting point to the goal, and in this respect is similar to most approaches to the problem of path planning for stationary obstacles. The problem which our path planner is designed to deal with, however, is one in which the scene description can change substantially and unpredictably from one epoch to the next Because of these changes, a global path would have to be recalculated every epoch, and such recalculation would be prohibitively expensive in terms of computing resources. This is why we have chosen the local approach mentioned above.

The path planner described here has potential applications either as a navigational-tactical aid for a manned vehicle, or as a component of the planner for an autonomous vehicle. As a possible sample scenario, consider a ship returning to force. In this scenario, the moving obstacles represent the counterdetection fields of various contacts to be avoided, while the moving goal region represents the ship's surface action group.

Our approach to path planning takes as its starting point the "maneuvering board" calculation technique commonly used in nautical navigation. In the next section, we describe the way in which we have used this maneuvering board approach as the basis for developing a path planner. In order to focus on the basic mechanism, we will assume at first that the path planner always has perfect knowledge of the instantaneous position and velocity of all objects in its environment. In later sections, we describe extensions to the basic mechanism which enable the path planner to deal with uncertainty in position and velocity.

2 Maneuvering Board Calculations

The basic maneuvering board mechanism is illustrated in Figures 1 and 2. As mentioned, we assume in these first examples that the planner has perfect knowledge of the position and velocity of all objects. In Figures 1 and 2, therefore, obstacles are represented as circles and the vehicle is represented as a point. For example, the circle

* Currently at AAI Corporation, Hunt Valley, MD

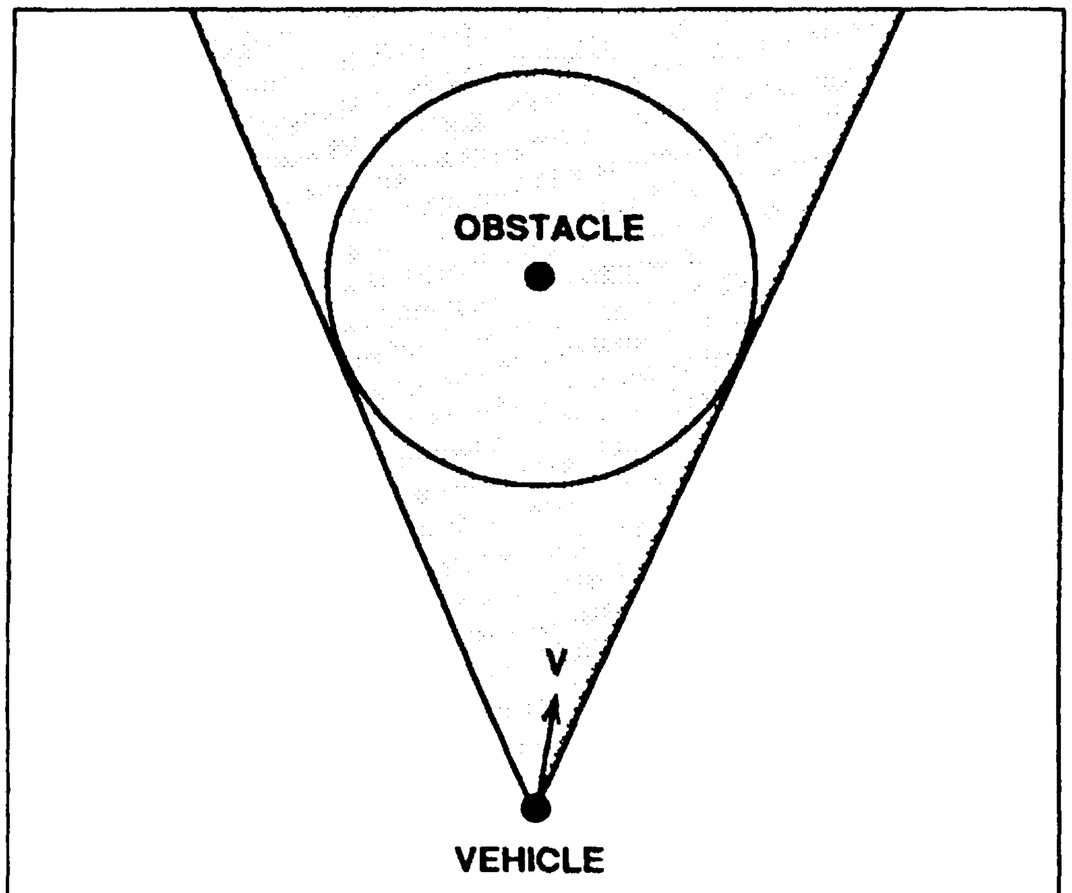

Figure 1: Avoidance cone for stationary obstacle

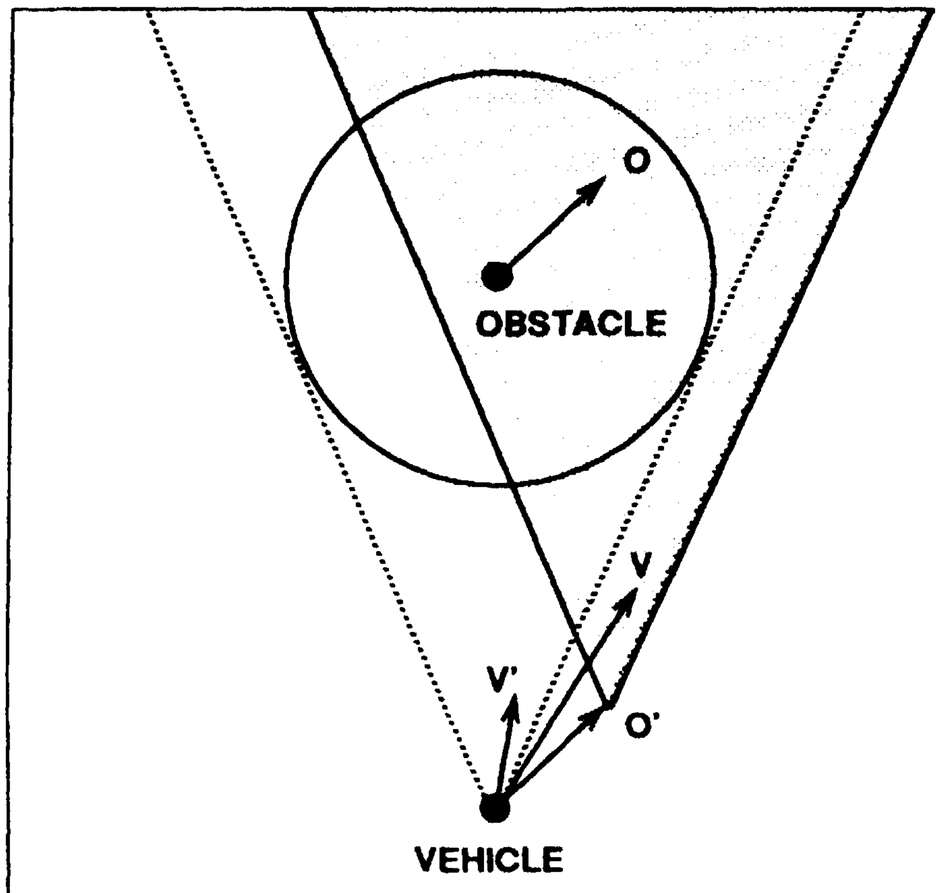

Figure 2: Avoidance cone for moving obsta

for an obstacle might represent the counterdetection zone of an enemy vessel located at the center.

Figure 1 shows the avoidance cone for a stationary obstacle. This cone is defined by the two tangent lines from the vehicle to the circle representing the obstacle. Clearly, any vector *V* which falls within this cone will put the vehicle on a collision course with the obstacle.

In order to construct the avoidance cone for a moving obstacle, we proceed as shown in Figure 2. First, translate the velocity vector *O* of the obstacle to the vehicle location (thus defining vector *0'* in Figure 2), and then translate the cone that would be generated in the stationary case to the tip of this translated vector. Any vector *V* for the vehicle that lies inside this cone represents a collision course with the moving obstacle. To see this, note that any such vector is the vector sum of a vector *V* which represents a collision vector if the obstacle were stationary, and *0* which intuitively "compensates" for the motion of the obstacle.

3 Determining a Course and Speed

At each epoch, we use the avoidance cones constructed for the current scene to determine an actual course and speed for the vehicle. We have employed a simple and efficient heuristic for choosing a course and speed. Informally, this heuristic can be described as follows: Define an optimal course for the given solution, and then choose a vector which is as close as possible to the optimal, while satisfying all constraints. More precisely, choosing a course and speed for the vehicle involves following a three-step procedure:

1. Determine the intercept point for the goal.
2. Generate the set of candidate vectors.
3. Choose the best candidate vector.

3.1 Determining the Intercept Point

The intercept point is defined by the point at which the vehicle would intercept the center of the goal, if the vehicle were to follow a direct intercept course at its maximum speed and the goal were to continue at its current velocity.

3.2 Generating the Set of Candidate Vectors

Let C be the set of cones constructed for the current scene, and let 5 be the set of line segments defined by the cones in C, together with the segment whose endpoints are the vehicle's position and the intercept point /.

(a) For each segment s in 5, determine all intersection points of *s* with {r : *r e S - [s]}* (i.e., all segments other than *s),* discarding all resulting points which lie outside the vehicle's maximum speed circle.

(b) For each *s* in 5, determine all intersection points of *s* with the maximum speed circle of the vehicle.

(c) The set of candidate points is the set of all points generated in step (a) or (b) which do not lie inside the avoidance cone of any obstacle.

This procedure is illustrated in Figures 3 and 4, where the circle about the vehicle represents the maximum speed of the vehicle in any direction. Note that the points generated in this way correspond to discernible strategies. In Figure 3, for example, point *P* represents a maximum speed, direct intercept course with the goal. In Figure 4, such a direct intercept course would result in a collision with an obstacle, and hence the corresponding candidate point is not generated. Instead, point *A* corresponds to a strategy of following a maximum-speed trajectory directed around and tangent

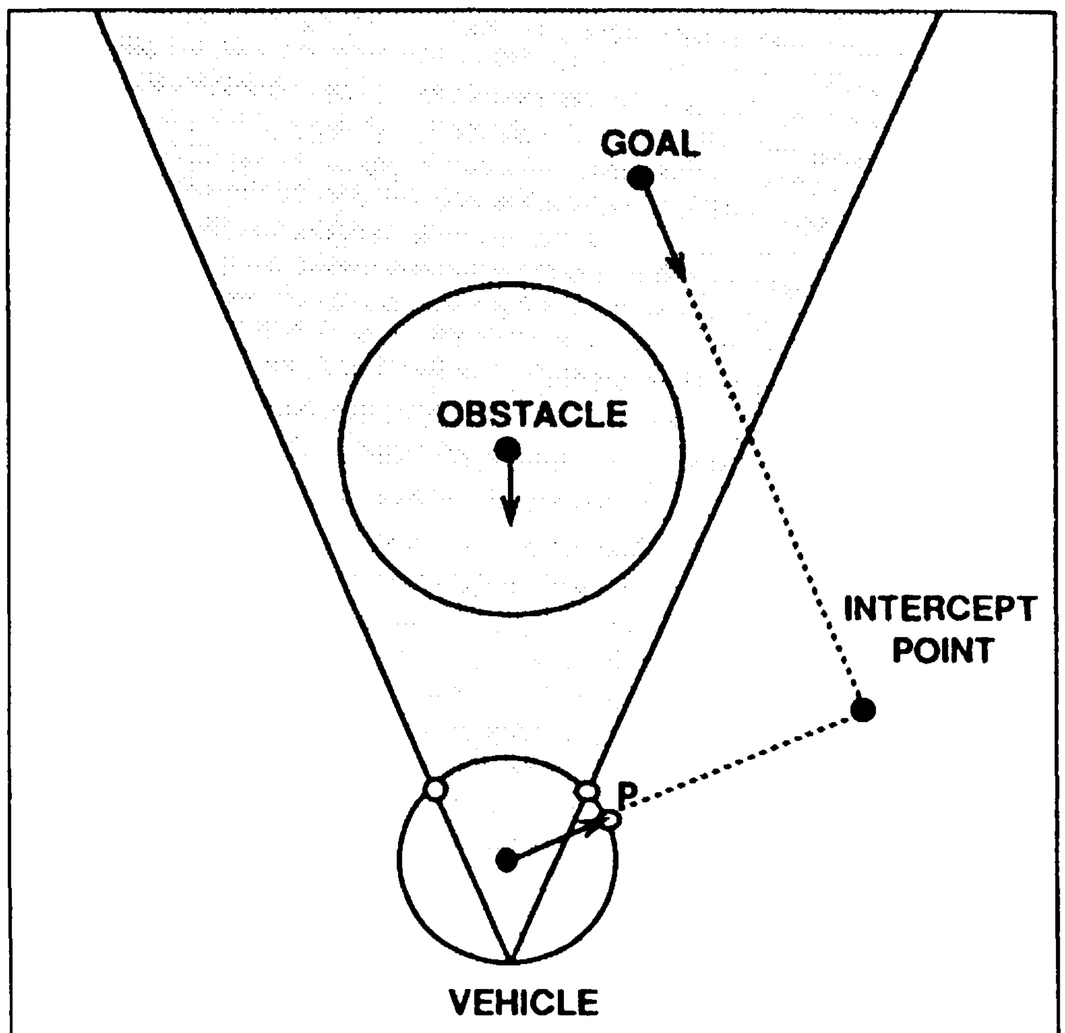

Figure 3: Determining course and speed (1)

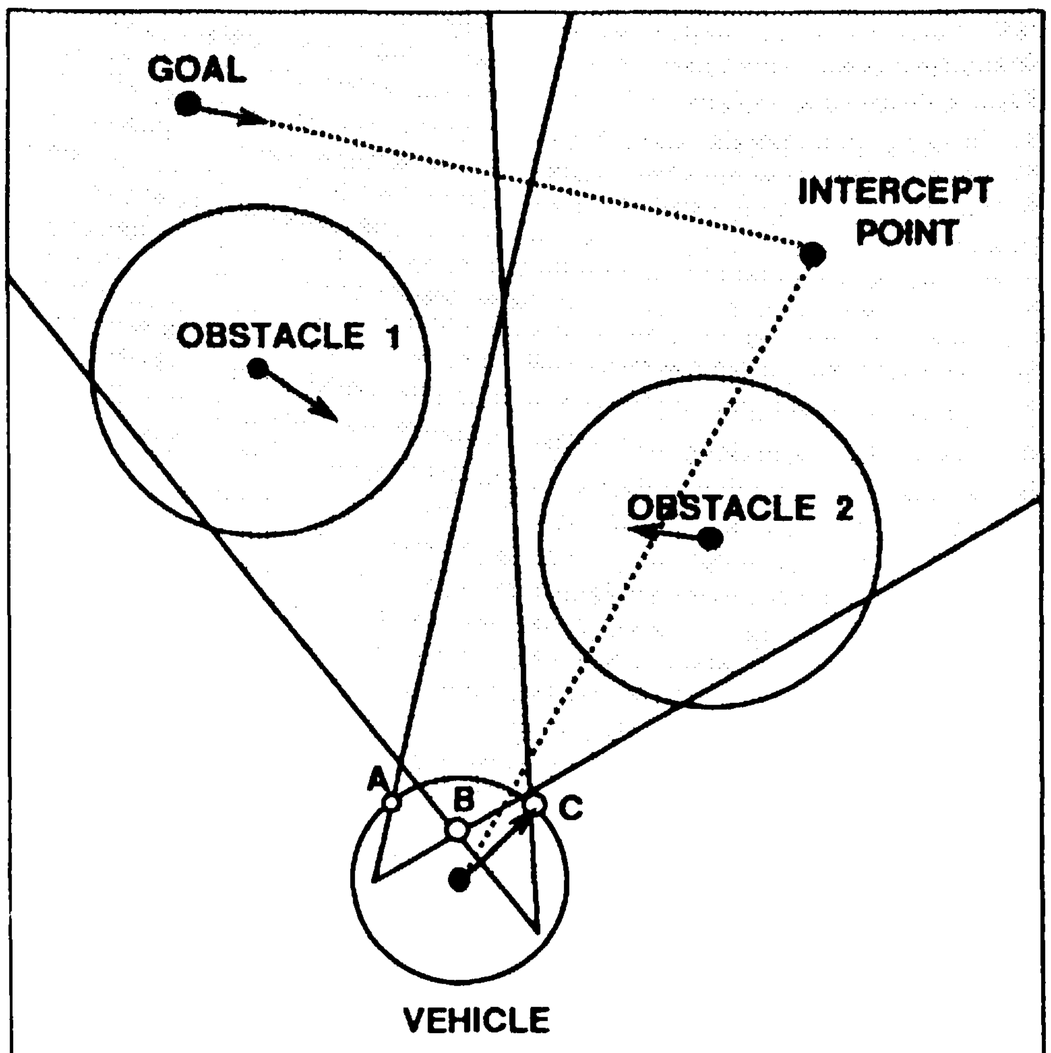

Figure 4: Determining course and speed (2)

to obstacle 1, and which is fast enough to avoid collision with obstacle 2. Point *B* corresponds to a strategy of moving slowly towards the goal until the two obstacles have passed one another, and then speeding up to reach the goal. Point *C,* the vector actually chosen by the method, corresponds to the strategy of following a maximum-speed trajectory which is directed around and tangent to obstacle 2, and which is fast enough to avoid collision with obstacle 1.

3.3 Choosing the Best Candidate Vector

We choose as the velocity vector for the vehicle the vector that corresponds to that candidate point which is closest (in Euclidean, straight-line distance) to the intercept point. Thus when a point which corresponds to the maximum speed, direct intercept course is available, as in Figure 3, the method simply chooses that point. When such a point is not available, the method chooses the closest alternative. In Figure 4, for example, the method chooses point *C.* The choice of point *C* over, say, point *A* makes sense intuitively because C, unlike *A,* represents a vector which points in the direction of motion of the goal.

As was stated earlier, this process of determining a best course and speed for the current situation is repeated each epoch. To illustrate how these isolated decisions combine to drive the vehicle through a field of obstacles toward its goal, Figures 5a - 5c show a temporal sequence. Note that while no look ahead was performed, the resulting path is close to optimal.

4 Dealing With Uncertainty in Position and Velocity

In order to represent uncertainty in the position of a given obstacle, we use the representation scheme shown in Figure 6. Here the inner ellipse represents the set of possible positions for the obstacle, and the outer ellipse represents the avoidance range for the points in the inner ellipse. However, the basic mechanism remains unchanged for this extension. We define avoidance cones just as we did before, except that we calculate tangents to the outer ellipse rather than to a circle. The resulting avoidance cones are used exactly as outlined in Section 3.

If the course and/or speed of the obstacle are uncertain, then the endpoint of the obstacle's vector is also represented as an ellipse e, as shown in Figure 7. In this case, the set *A* of points to be avoided is constructed as follows. Let *O* be the velocity vector for the obstacle at the center of e, *c* the cone that would be constructed from 0, *0'* the result of translating *0* to the vehicle's position, and *e* the result of translating *e* to the vehicle's position. Then *A* is the set of points swept out by translating *c* about *e'*. In order to work with a geometric structure which is computationally more tractable than the set *A,* we approximate this set using the following procedure. First, parallel translate the two sides of *c* until they are tangent to *e* Then define the base of the new structure by constructing a line segment which joins the translated sides, is tangent to *e* and is perpendicular to a line from the vector endpoint of the obstacle to the vertex of the original cone. The result of this procedure is shown in Figure 7. Again, the basic mechanism described in Section 3 is applied unchanged to these truncated cones.

5 Time Truncation

The maneuvering board mechanism, as described here, represents a simple and efficient approach to the problem

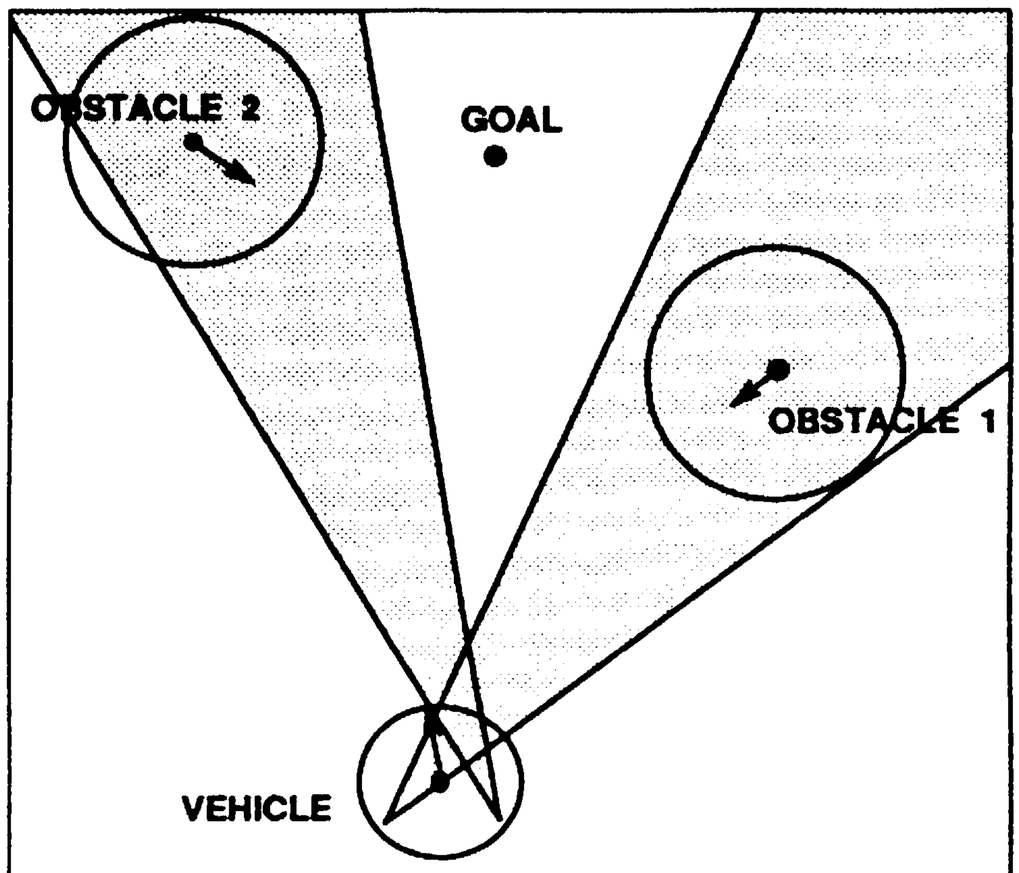

Figure 5a: Epoch 1 - Initial State

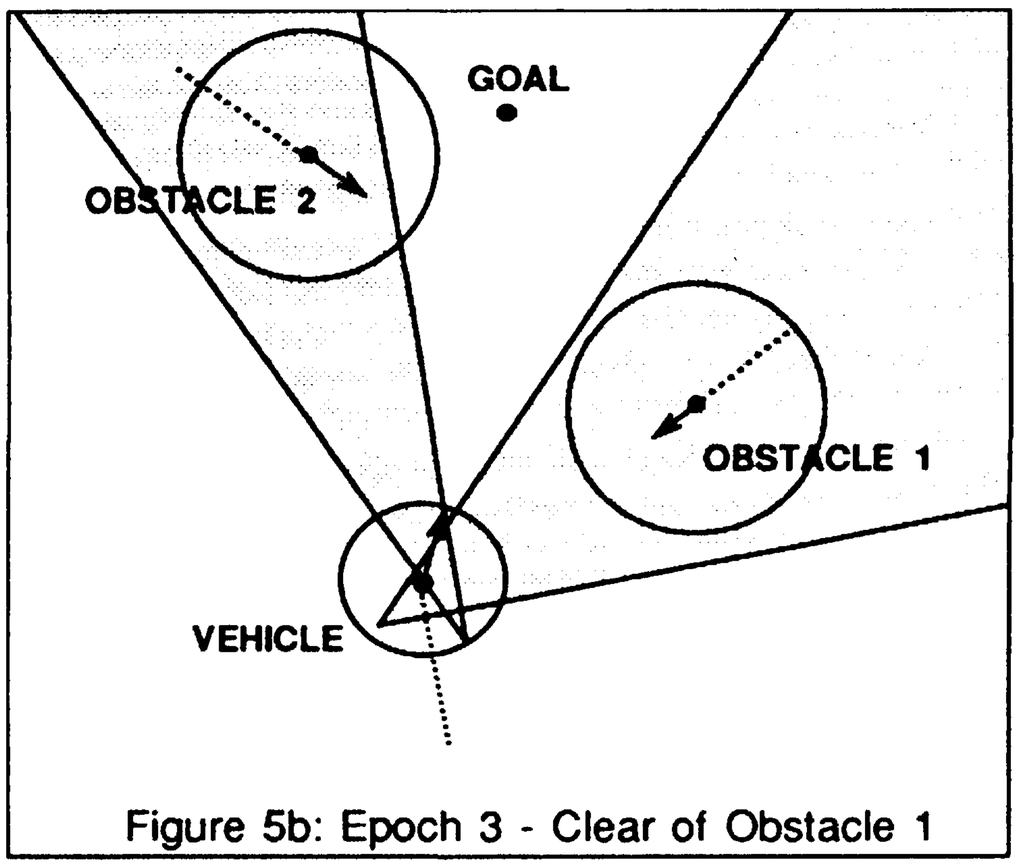

Figure 5b: Epoch 3 - Clear of Obstacle 1

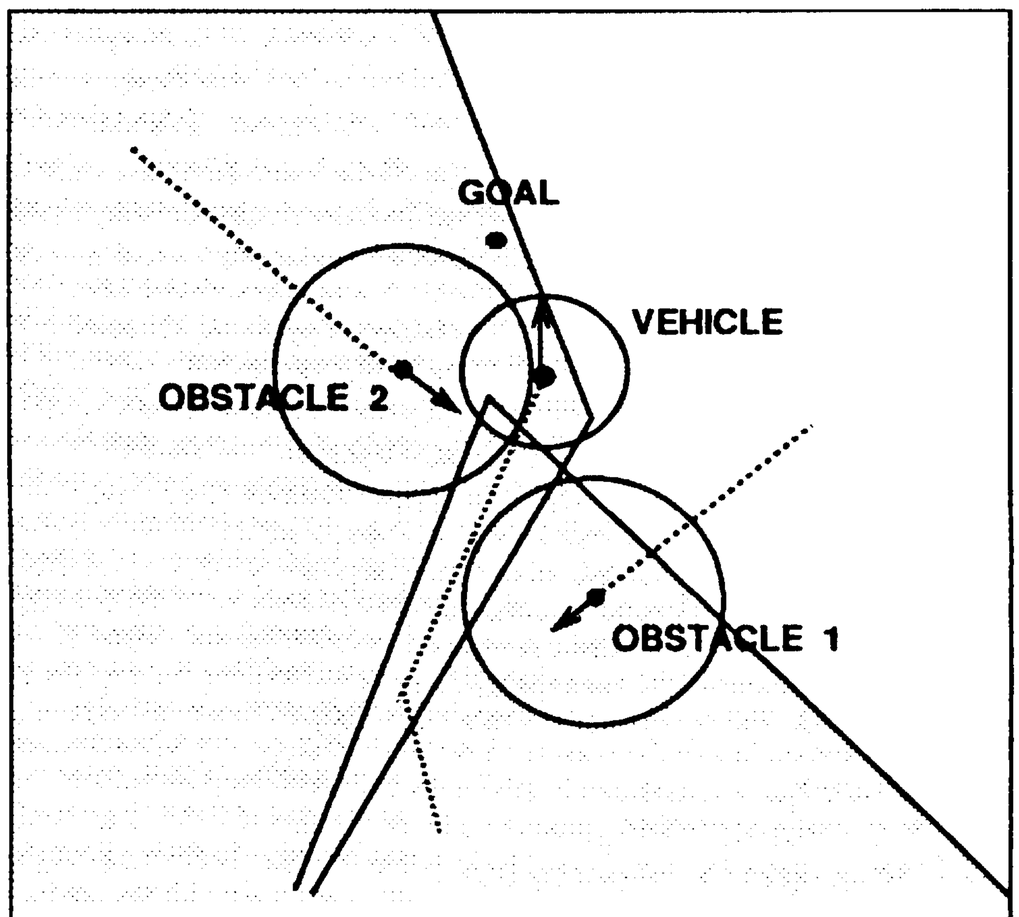

Figure 5c: Epoch 7 - Clearing Obstacle 2

of finding a collision-free path through a field of moving obstacles. In particular, by rejecting all potential velocity vectors for the vehicle which lie inside of an obstacle avoidance cone, we guarantee that the vehicle will follow a collision-free path. Of course, the requirement that any velocity vector for the vehicle lie outside of all avoidance cones represents only a sufficient, and not a necessary condition for finding a collision-free path. The requirement does not represent a necessary condition because it does not take into account the time to potential collision between the vehicle and various obstacles. In other words, while it is true that any vector which lies inside an avoidance cone represents a collision course with the corresponding obstacle, this means only that the vehicle will eventually collide with the obstacle if both continue for a long enough time in their present states of motion. But there is no guarantee that the obstacle will continue indefinitely in its present state of motion; and even if it does, the eventual collision might not take place until long after the vehicle has reached its goal.

In order to take time into account, we allow an optional specification of the minimum number of epochs for which the recommended course and speed must avoid collision. This is accomplished by truncating the avoidance cones, using a method which will be explained below. For example, if the specified number of epochs is five, then any vector which is outside of the resulting avoidance cones would be guaranteed to avoid collision for at least five epochs.

When a minimum number of epochs has been specified, we truncate the cone associated with any obstacle by an amount which is inversely proportional to the specified number of epochs. More precisely, let *e* be the number of epochs, and *d* the distance from the base of an avoidance cone to the outer ellipse for the corresponding obstacle; thus *d* is the length of a velocity vector which leads the vehicle to collide with the obstacle in one epoch. Then the original cone is truncated by *die,* as shown in Figure 8. A velocity vector for the vehicle which falls outside of the truncated one is guaranteed to avoid collision for at least *e* epochs.

6 Barrier Detection Critic

One of the problems with the closest to intercept strategy used by the path planner is that the vehicle can get trapped by a "barrier" formed by a concave configuration of obstacles. The problem is caused by the fact that in order to get around the barrier, the vehicle must actually move away from its destination. Such motion is contrary to the closest to intercept strategy. Consequently, the path planner finds that it can stay closer to its destination by sitting still inside the trap than by going around the trap.

A possible solution to this problem which we are currently experimenting with is to add a "critic" to the path planner which detects barriers and rejects possible moves which would lead the vehicle into a trap.

Any set of obstacles which intersect and which are

between the vehicle and its destination are identified by the critic as a barrier. Once all barriers have been identified, the critic is able to determine for each potential motion generated by the path planner whether it moves the vehicle into a trap with respect to any of the barrier. This is done by determining the geometric center of each barrier (analogous to the center of mass). A line is then drawn from the destination through the center of the barrier. Any vector which moves the vehicle closer to this line is deemed unacceptable by the critic, because it is potentially leading the vehicle into a trap.

The path planner follows the critic's evaluation as long as there is at least one potential move which the critic finds acceptable. If all of the available moves are deemed unacceptable by the critic, then the path planner disregards the critic and makes what it considers to be the best move.

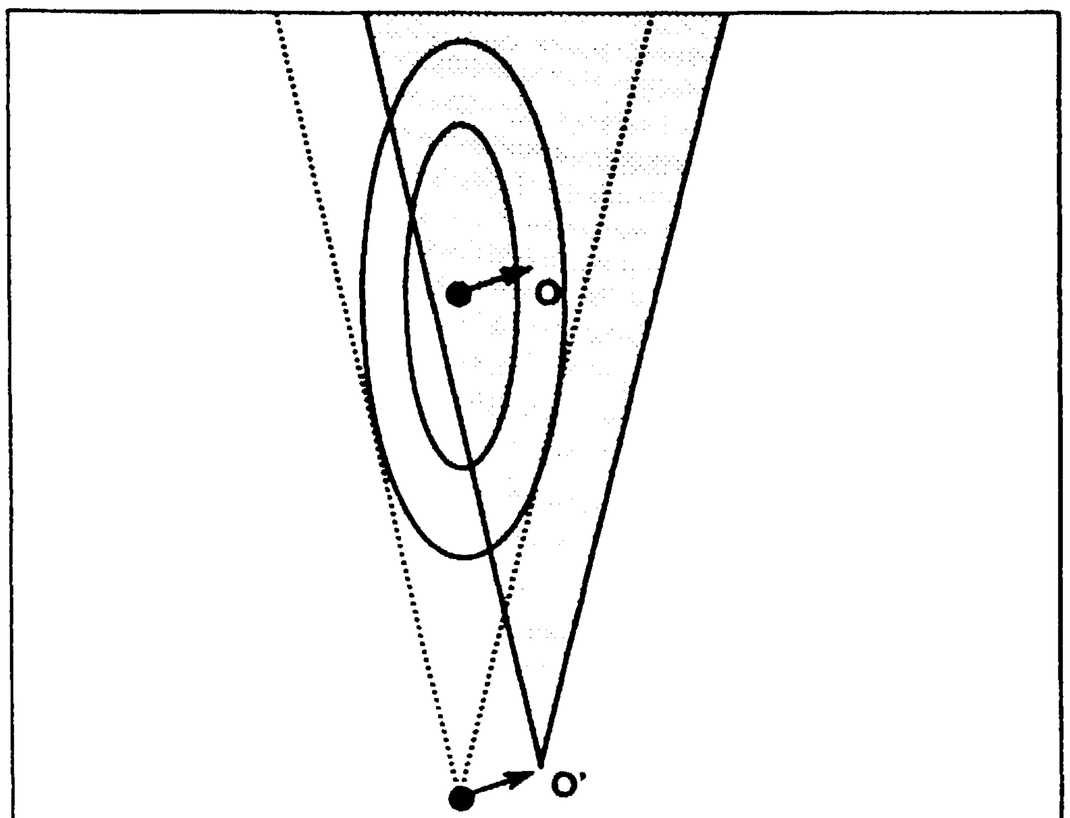

Figure 6: Positional Uncertainty

7 Implementation

The path planner has been implemented in Common Lisp on both Symbolics and Sun workstations. All entities are represented within an object-oriented paradigm, using the object-oriented programming tool FROBS (FRames and OBjectS) [Muehle, 1987]. FROBS was chosen over other Lisp-based object-oriented tools such as Flavors because it is portable to any other Common Lisp.

Processing time for the algorithm is proportional to the number of candidate points generated. In the worst case, each truncated avoidance cone intersects the vehicle's maximum speed circle at 6 points, and two such avoidance cones intersect each other at 6 points. Hence the number of candidate points generated is, in the worst case, $3N(N + 1) = 0(N^2)$, where N is the number of obstacles.

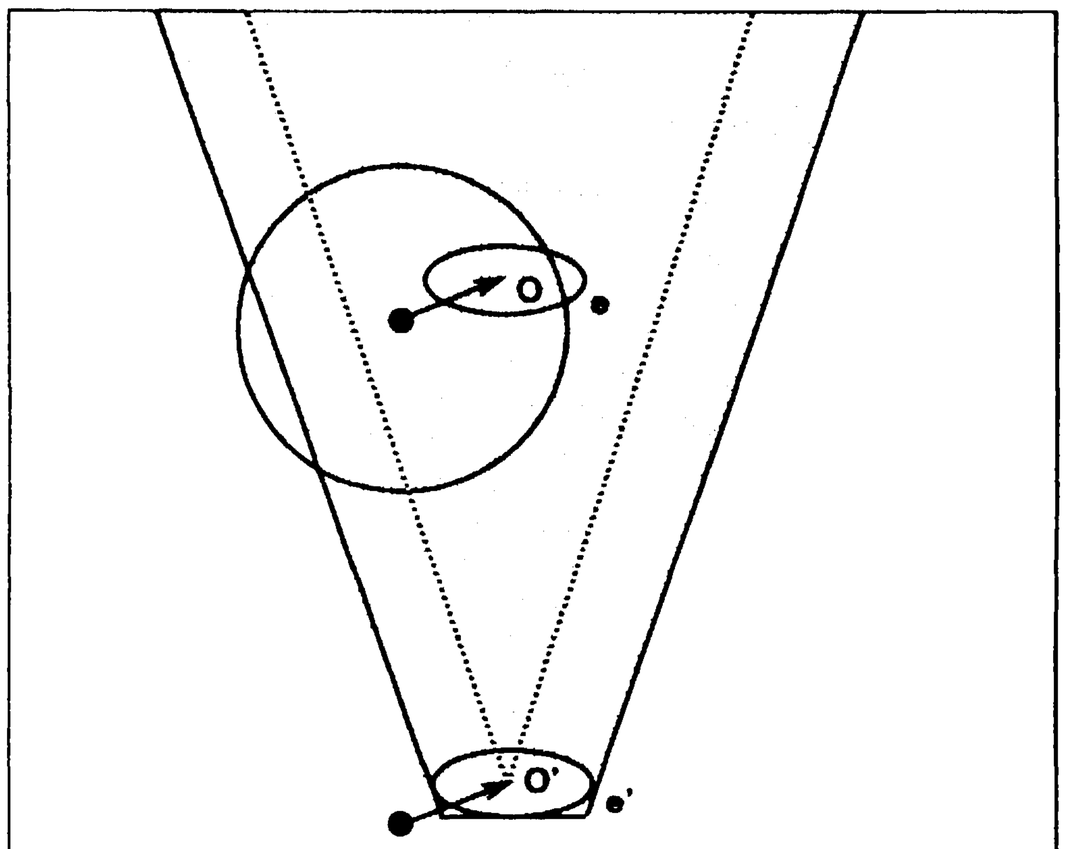

Figure 7: Course and Speed Uncertainty

References

[Brooks, 1983] R. Brooks. Solving the find-path problem by good representation of free-space. *IEEE Transactions on Systems, Man, and Cybernetics,* SMC-13, 190-197, 1983

[Lozano-Perez] T. Lozano-Perez. Spatial planning: a configuration space approach. *IEEE Transactions on Computers,* C-32 2, 108-117, 1983

[Lozano-Perez and Wesley] T. Lozano-Perez and M.A. Wesley. An algorithm for planning collision-free paths among polyhedral obstacles. *Communications of the ACM* 22, 560-570,1979.

[Muehle, 1987] E. Muehle. *FROBS User Guide,* University of Utah PASS Group, 1987.

[Reif and Sharir] J. Reif and M. Sharir. Motion planning in the presence of moving obstacles. *Annual Symposium on Foundations of Computer Science,* 144-154,1985.

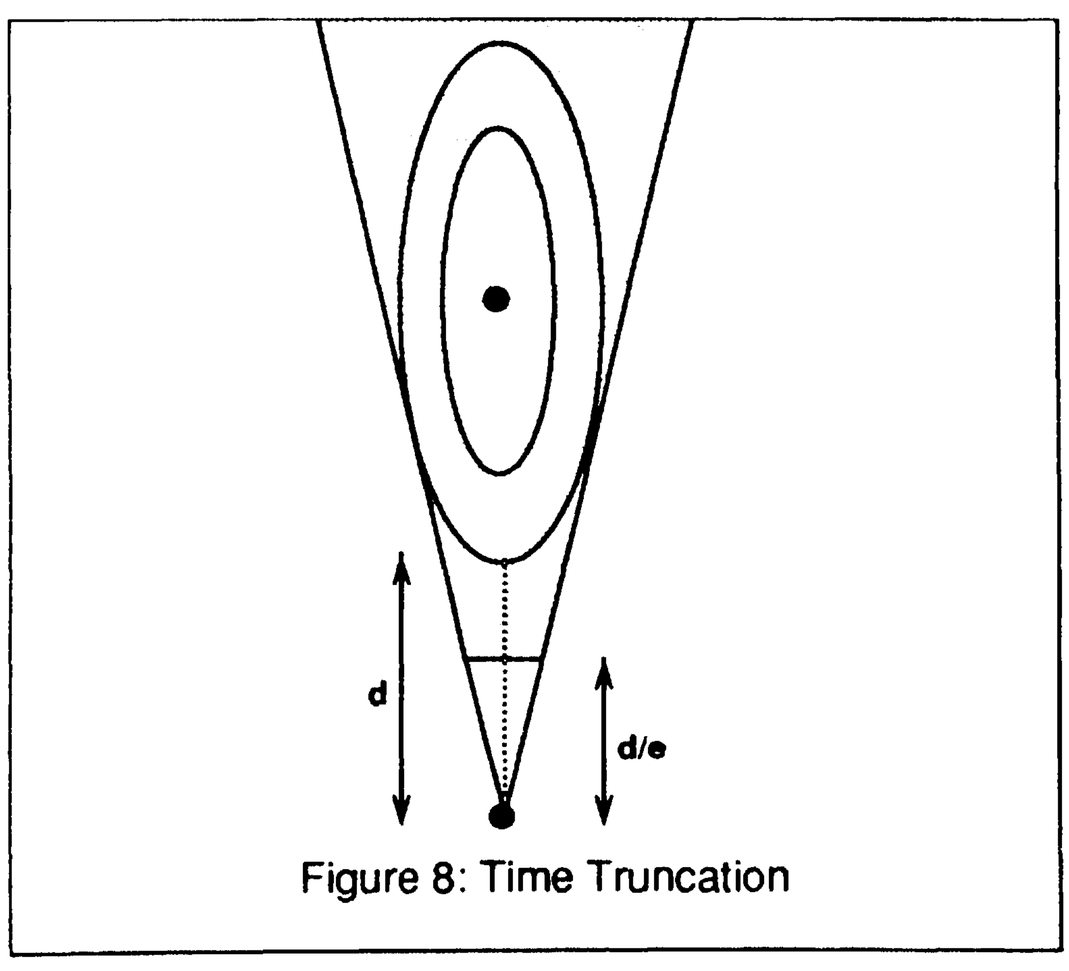

Figure 8: Time Truncation