A Robot Navigation Algorithm for Moving Obstacles

Lynne E. Parker

To the Graduate Council:

I am submitting herewith a thesis written by Lynne E. Parker entitled "A Robot Navigation Algorithm for Moving Obstacles." I have examined the final electronic copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Computer Science.

Robin Cockett, Major Professor

We have read this thesis and recommend its acceptance:

Accepted for the Council:
Carolyn R. Hodges

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

To the Graduate Council:

I am submitting herewith a thesis written by Lynne E. Parker entitled "A Robot Navigation Algorithm for Moving Obstacles". I have examined the final copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Computer Science.

Robin Cockett

Robin Cockett, Major Professor

We have read this thesis
and recommend its acceptance:

[illegible]

[illegible]

Accepted for the Council:

[illegible]

Vice Provost
and Dean of The Graduate School

STATEMENT OF PERMISSION TO USE

A ROBOT NAVIGATION ALGORITHM FOR MOVING OBSTACLES

A Thesis

Presented for the

Master of Science

Degree

The University of Tennessee, Knoxville

Lynne E. Parker

June 1988

ABSTRACT

In recent years, considerable progress has been made towards the development of intelligent autonomous mobile robots which can perform a wide variety of tasks. Although the capabilities of these robots vary significantly, each must have the ability to navigate within its environment from a starting location to a goal without experiencing collisions with obstacles in the process — a capability commonly referred to as "robot navigation".

Numerous algorithms for robot navigation have been developed which allow the robot to operate in static environments. However, little work has been accomplished in the development of algorithms which allow the robot to navigate in a dynamic environment. This thesis presents a mathematically-based navigation algorithm for a robot operating in a continuous-time environment inhabited by moving obstacles whose trajectories and velocities can be detected.

In this methodology, the obstacles are represented as sheared cylinders to depict the areas swept out by the obstacle disks of influence over time. The robot is represented by the cone of positions it can reach by traveling at a constant speed in any direction. The methodology utilizes a three-dimensional navigation planning approach in which the search points, or tangent points, are the points in time at which the robot tangentially meets the obstacles. These tangent points are determined by calculating the intersection curves between the robot and the obstacles, and then using the first derivative of the intersection curves to make the tangent selections. Paths are created as sequences of these tangent points leading from the robot starting location to the goal, and are searched using the A* strategy, with a heuristic of the Euclidean distance from the tangent point to the goal.

The main contribution of this thesis is the development of a methodology which produces optimal tangent paths to the goal for a dynamic robot environment. This feature is significant, since no other algorithm located in the literature survey as background to this thesis has shown the ability to produce paths with optimal properties.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 APPLICABILITY/NEED FOR ROBOT NAVIGATION ALGORITHMS

Significant strides have been made in recent years towards the development of intelligent autonomous robots which can perform a wide variety of tasks, from mail delivery to explosive ordnance disposal. The potential benefits of such robots have contributed to the burgeoning interest in their development, since robots can increase productivity and relieve humans of dull, repetitive, or dangerous tasks. The capabilities of these state-of-the-art robotic systems vary from robot to robot, but often include subsystems for image processing or vision, learning, intelligent reasoning, sensing, and manipulation.

In order for these intelligent robots to operate, however, they must have the capability to navigate within their environment, a capability commonly referred to as "robot navigation". The mobile robot must be able to travel from a starting location to a goal location in the environment without experiencing collisions with obstacles or barriers in the process. The development of a robot navigation algorithm is not a trivial matter, since it requires decisions on a number of critical issues, such as the type and representation of the robot and obstacles, the type of world model to be utilized, the extent of the knowledge available about the robot environment, and the characteristics of the resulting robot path.

1.2 PURPOSE

The goal of this thesis is to develop a mathematically-based robot navigation algorithm which operates in a continuous-time environment inhabited by moving obstacles whose trajectories and speeds can be detected. The need for such an algorithm should be apparent from the review of existing methodologies in the following chapter, as not much work has been accomplished in this area. Consider an autonomous mobile robot which is trying to cross a street or make its way through a room occupied with people or other mobile robots. Ideally, the robot should behave as a human would behave in this situation — it would note the current locations, speeds, and trajectories of the moving objects and plan its route to the goal based upon the projected locations of the obstacles through time.

Many aspects are involved in providing a robot with the capabilities required to accomplish such a task. The robot must have some means to sense the important features of its environment, meaningfully interpret the sensed data, plan a feasible navigation route based on the interpretations, execute its navigation plan, and successfully monitor the execution of the plan. Each of these areas is a research topic in itself with many difficult issues to be addressed. The present thesis, however, will consider only the navigation planning subject based upon the interpreted sensor data, as much can be contributed to this facet of the problem.

The methodology developed in this thesis applies to a robot which can move at a constant speed in an environment occupied by obstacles moving at constant velocities. In a real-world environment, the execution of the navigation plan would be closely coupled with continual environmental sensing to detect changes in the speed or trajectory of the obstacles for plan modification or total replanning if required. Additionally, extensions to the methodology can provide a means to relax the constant robot speed assumption. Refer to chapter 6, for more details on this extension.

1.3 APPROACH

The methodology developed in this thesis uses a configuration space approach [5, 22, 30, 32, 42, 43, 54, 60, 61, 62, 84, 90] in which the obstacles, approximated by their bounding circles, or disks of influence, are assumed to be "grown" such that the resulting representation corresponds to all possible undesirable interactions between the obstacles and the robot in two-dimensional Cartesian space. A third dimension is added to the representation to incorporate time into the planning process. The obstacle then becomes a sheared cylinder representing the area swept out by the obstacle disk of influence over time. The robot is represented by the cone of positions it can reach by traveling at a constant speed in any direction. These representations are particularly suited for dealing with moving obstacles.

The methodology utilizes a three-dimensional navigation planning approach analogous to the two-dimensional visibility graph used for stationary obstacles [42, 60, 61, 62] which is described in the next chapter. Rather than the obstacle vertices being the search points of the visibility graph, the points in time at which the robot tangentially meets the obstacles are the search points, or tangent points. These tangent points are determined by calculating the intersection curves between the robot and the obstacles, and then using the first derivative of the intersection curve to make the tangent selection. Using the A* search with a heuristic of the Euclidean distance from the tangent point to the goal, a visible tangent point with the least cost is selected and a determination is made concerning the visibility of the goal. If the goal is visible, a new path is created from the current tangent point to the goal, its cost is calculated, and the path is inserted into the set of possible paths in ascending order according to cost. If the goal is not visible, the selected tangent point is expanded to find the next set of visible tangent points, and new paths are created corresponding to the newly detected tangents. This cycle of selecting a visible tangent with the least cost and expanding that tangent is repeated until the goal can be reached. Since the A* heuristic uses an estimate of the remaining distance to the goal which is a lower bound on the actual distance, this navigation algorithm will produce an optimal tangent path to the goal for the constant robot speed [89, pg. 115]. The ability to produce optimal tangent paths for a constant robot speed is significant, since no other algorithm located in the literature survey as background to this thesis has the ability to produce paths with optimal properties.

The following chapter provides an overview of existing robot navigation algorithms for both static and moving obstacles. Chapter 3 describes the details of the planning methodology while chapter 4 presents the overall path planning algorithm. Chapter 5 gives details on the results of the implementation of the algorithm, and chapter 6 contains concluding remarks.

CHAPTER 2

EXISTING ROBOT NAVIGATION ALGORITHMS

The environments in which robots perform their assigned tasks vary widely from the organized and predictable to the extremely unstructured and uncertain. The more unpredictable the environment is, the more dependence the robot's navigation control must place on sensor data, and the less certain the environmental model will be. Due to this difference in the characteristics of the environment, the navigation approaches employed in the various robot surroundings range from provable algorithmic methodologies to unprovable, although quite competent, heuristic approaches. In either case, however, the robot must have some idea of the world, and then plan based upon its environmental model.

The following sections will summarize the diverse methods which have been developed for solving the navigation problem. The large majority of the path planning algorithms were developed to pertain exclusively to surroundings in which the obstacles are stationary. These algorithms are reviewed in section 2.1. The relatively few algorithms for dynamic environments are considered in section 2.2, along with suggestions for the extension of the static environment algorithms to deal with moving obstacles. Refer to the paper by S. H. Whitesides [84] for a more detailed survey of much of the work which has been accomplished in this area, and to the paper by J. H. Reif [64] for an analysis of the computational complexity of the robot navigation problem.

2.1 STATIC ENVIRONMENT

2.1.1 Algorithmic Approaches. One large class of robot navigation algorithms requires detailed descriptions of the obstacles in the environment, and are usually variants of a specific navigation problem called the "FINDPATH" or the "Piano Mover's" problem. The general definition of the FINDPATH problem can be stated as follows [84]: Given the descriptions of the obstacles in a space, and the initial and final configurations of an object (e.g. a robot) in that space, determine whether there is a continuous transformation from the initial to the final configuration, and what that transformation should be. Individual algorithms often place additional restrictions on the problem scenario, such as the shape of the obstacles or of the moving object, whether the moving object is permitted to consist of moving parts, or whether the modeled environment is two or three dimensional.

The following sections describe existing algorithms to solve the FINDPATH problem. Section 2.1.1.1 describes algorithms adhering to the configuration space approach, whereas section 2.1.1.2 describes the Cartesian space approaches.

2.1.1.1 Configuration Space Approaches. One approach most prevalent in the existing FINDPATH algorithms is that of transforming the representation of the world to allow the robot to be treated as a point. Rather than planning with the Cartesian space symbolizing the actual physical problem, a configuration space is used. The configuration space is constructed by using a "growing" operation on the obstacles to yield a representation in configuration space which corresponds to all possible collisions between the obstacle and the robot in Cartesian space. This transformation reduces the task of calculating a collision-free path for a complex geometrical object to that of finding a safe path for a point object. An unfortunate disadvantage to this approach is that the simplification may be obtained at the cost of not allowing the robot to translate or rotate in the environment, or of over-restricting the robot motion.

Once the configuration space is obtained, several strategies for computing the robot path through the environment are possible. The major strategies utilized by existing algorithms are the visibility graph, the Voronoi diagram, and the cell segmentation approaches.

One of the earliest published works on the navigation of a robot through a known environment is the paper by T. Lozano-Perez and M. A. Wesley [42], which uses the visibility graph approach. This approach capitalizes on the fact that the shortest path from a starting location to a goal amidst polygonal obstacles in two dimensions will be composed of a sequence of straight lines connecting a possibly empty series of obstacles vertices. The resulting path plan causes the robot to travel from one obstacle vertex to another until the goal is reached. Specifically, the undirected visibility graph VG(N,L) is defined such that the node set N consists of the set of all vertices of the polygonal obstacles and the link set L is the set of all links (n_i, n_j) such that a straight line connecting the ith element of N to the jth element does not overlap any obstacles. The shortest collision-free path in two dimensions will be composed of a sequence of straight lines from VG(N,L) which join the starting location to the destination. The visibility graph for the sample environment shown in figure 1 is given in figure 2.

Once the visibility graph is constructed, various search methodologies are used to find the robot path. Lozano-Perez and Wesley [42] use an A* algorithm with a heuristic which estimates the cost to travel from a node to the goal, whereas

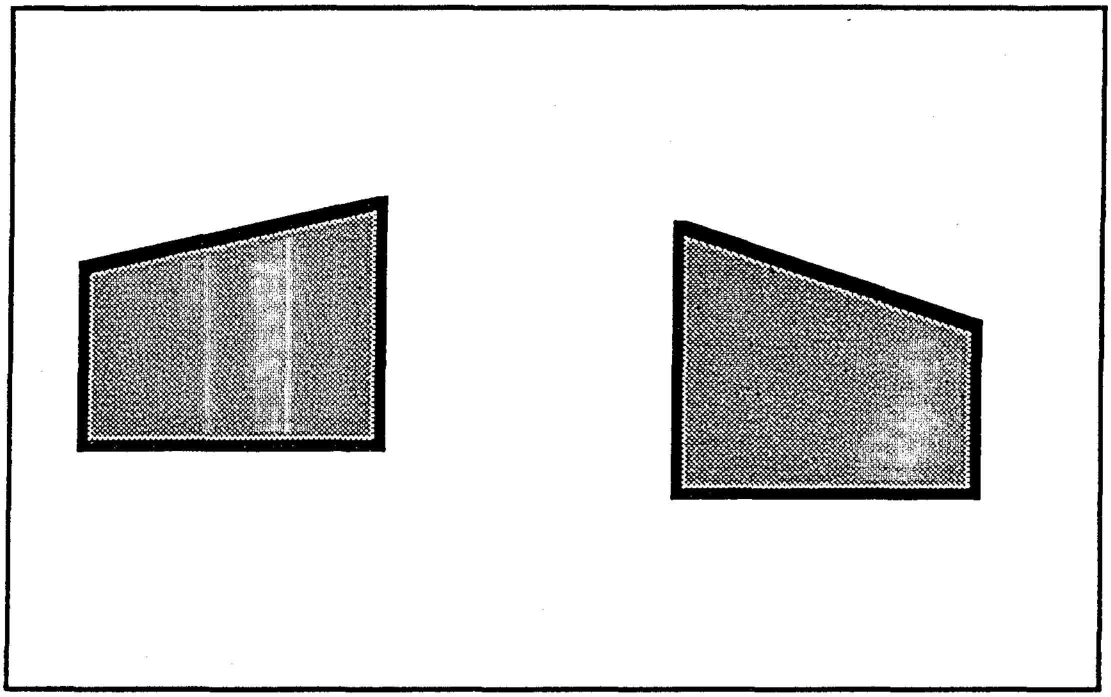


Figure 1. Sample robot environment.

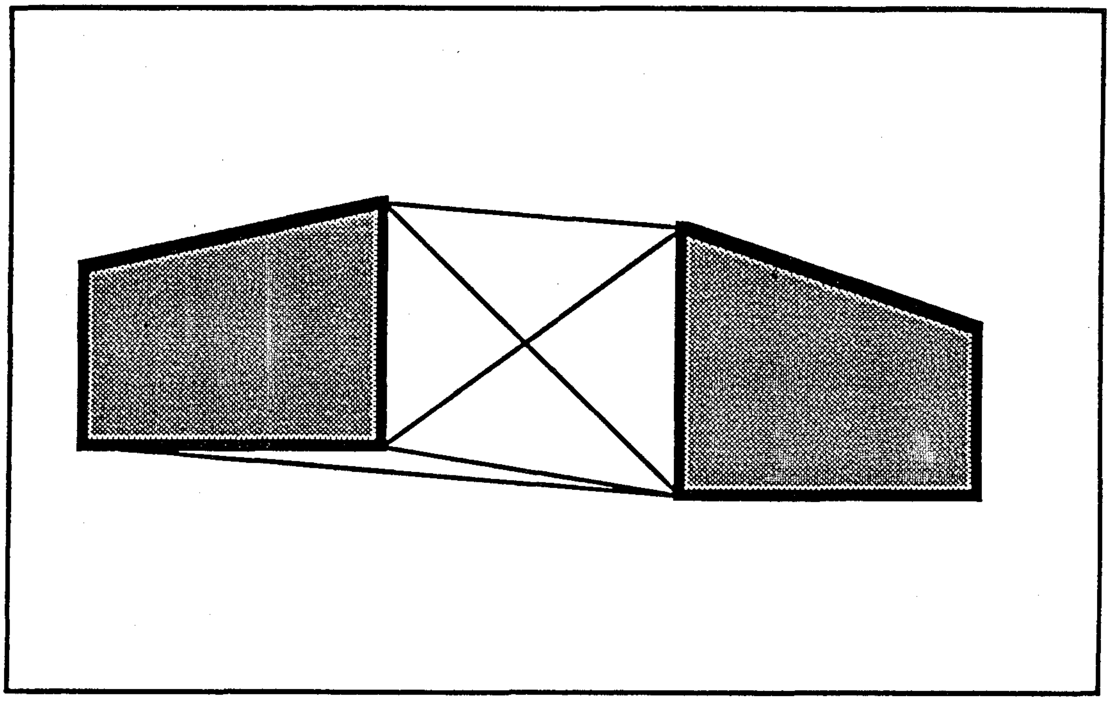


Figure 2. Visibility graph for sample robot environment.

N. Rao, et al. [60, 61, 62] use a depth first search of the graph to find the path to the goal.

An advantage of the visibility graph approach is that it will produce optimal paths in two dimensions. However, it suffers from the disadvantage that the resulting path hugs the obstacles, making the methodology prone to uncertainties in the environmental model and causing the monitoring of the robot path execution to be difficult. Unfortunately, the visibility graph approach will not produce optimal paths with a three-dimensional world representation, since the optimal path will not necessarily follow the vertices of the polyhedra. Lozano-Perez [42] suggests adding additional vertices along the edges of the grown obstacles such that no edge is longer than a pre specified length. This approach allows the algorithm to produce a good approximation to the optimum path.

A second major strategy for searching through configuration space uses the idea of a Voronoi diagram [30,54]. The paths derived from the Voronoi diagram will be inherently different from those obtained from the visibility graph. The paths derived using the Voronoi diagram will cause the robot to remain as far from the obstacles as possible, while the paths obtained from the visibility graph will take the robot very close to the obstacles. Due to this characteristic, the Voronoi diagram approach will rarely produce optimal paths. On the other hand, this trait will cause the approach to be less sensitive to errors in the world model and will usually simplify the monitoring of the path execution.

The standard Voronoi diagram [54, 84] is a planar network in which S is defined to be the closed set of polygonal obstacles in a plane whose complement is bounded. The Voronoi diagram for S, VOR(S), is the set of all points in the complement of S which have more than one obstacle as their closest neighbor. The search for a safe robot path is thus converted to finding a safe path in the

Voronoi diagram. Figure 3 depicts the Voronoi diagram for the environment shown in figure 1.

The third major strategy for searching the configuration space for a safe path uses a different philosophy than the visibility graph or the Voronoi diagram approaches, since the representation of the obstacles' complement, the "free space", is the primary search focus. This method, the cellular decomposition approach, involves dividing the configuration space into cells which represent the collision-free locations of the moving robot. A graph is then constructed with vertices representing the cells and the edges representing their adjacencies. As in the visibility graph and Voronoi diagram approaches, the path planning is thus transformed into a graph search.

Several variations to this basic theme have been developed [5, 22, 32, 43, 90]. S. Kambhampti and L. S. Davis [32] define a quadtree structure consisting of free nodes, obstacle nodes, and gray nodes. The free nodes represent regions of free space, the obstacle nodes represent regions containing obstacles, and gray nodes represent a mixture of free space and obstacles. The quadtree itself is a recursive decomposition of a two-dimensional picture into uniformly "colored" 2^i x 2^i blocks, where i is a positive integer. A node of the quadtree represents a 2^j x 2^j, $j < i$, square region of the picture. The leaf nodes of the quadtree are always either obstacle nodes or free nodes. Figure 4 gives an example of a portion of the quadtree for the environment shown in figure 1.

Planning a path from a starting point to a goal location first involves locating the leaves of the quadtree, S and G, which contain the starting and goal locations. A minimum cost path between S and G is then formed with the non-obstacle leaf nodes of the quadtree using the A* search with a heuristic representing the estimated cost of the remaining path to the goal.

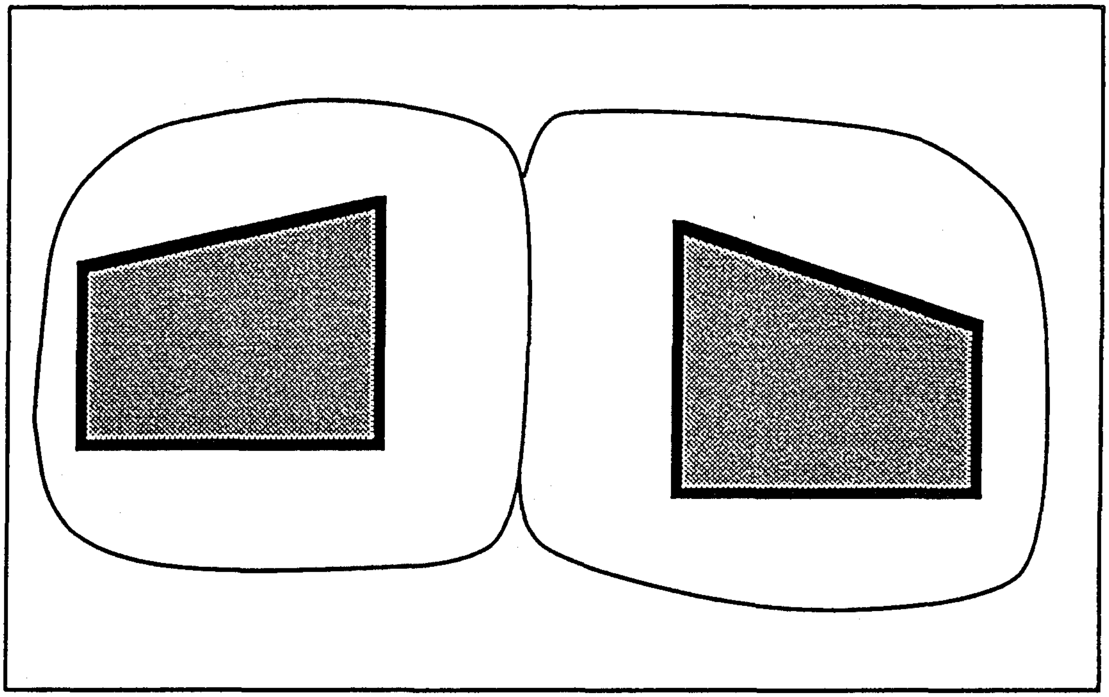


Figure 3. Voronoi diagram for sample robot environment.

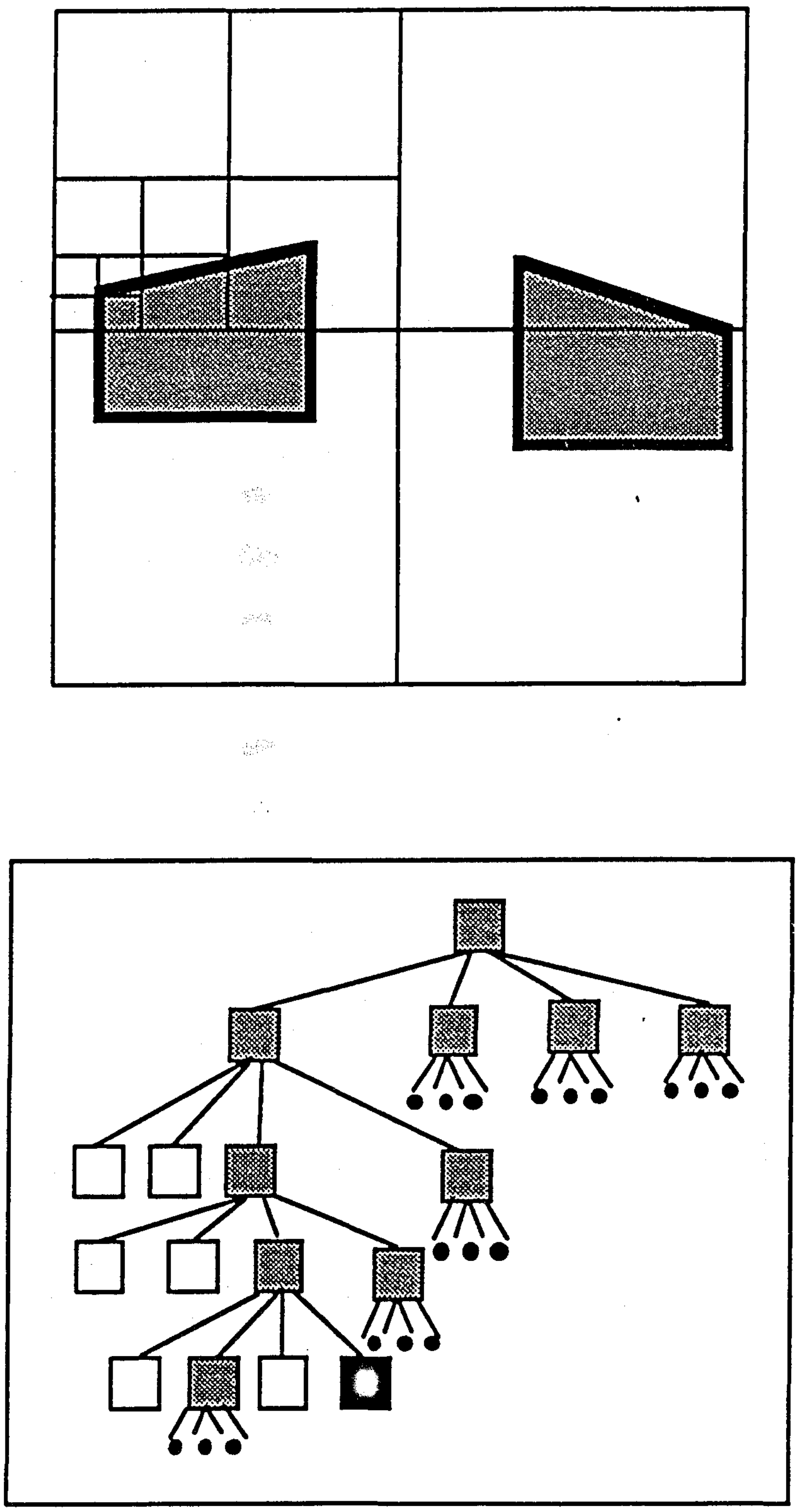


Figure 4. Quadtree representation for sample robot environment.

E. K. Wong and K. S. Fu [90] present a slight modification to the quadtree which allows free nodes and obstacle nodes to also have children. Lozano-Perez and Wesley [42] utilize a hybrid cell representation involving rectangular solids in areas distant from obstacles for reduction of detail, and convex polyhedra near obstacles where high accuracy is required. Each cell is labeled EMPTY, FULL, or MIXED, in a manner analogous to the full, obstacle, and gray nodes used by Kambhampti and Davis. Brooks and Lozano-Perez [5] present another variation using only rectangular solid cells.

The advantage of the cellular decomposition approach is that the search space is reduced by examining the free space nodes first. Only when this approach fails are the children of the gray (or MIXED) nodes searched. Additionally, this method allows paths to remain away from obstacles except when absolutely necessary. The disadvantage of this approach is that the representation does not localize the effect of obstacles and the resulting path will not typically be the shortest path.

2.1.1.2 Cartesian Space Approaches. Numerous additional algorithms operate directly in Cartesian space, rather than in configuration space. These methodologies often deal with more complex robot shapes which are not amenable to an approximation as a point. Perhaps the most important approaches developed in this category are the artificial potential field and the generalized cones methodologies.

The artificial potential field method, developed by Khatib [33], offers a significantly different approach to the path planning problem. The philosophy behind the artificial potential field approach is to visualize the robot as moving in a field of forces. The goal to be reached is an attractive pole while the obstacles are repulsive surfaces for the robot. The artificial potential field of an obstacle will be a non-negative continuous and differentiable function whose value approaches infinity as the robot approaches the surface of the obstacle. The collision-free path of the robot is determined by linking the absolute minima of the potential fields.

The potential field concept is an interesting approach to the path planning problem. However, the complexity of the tasks which can be solved by this approach is limited due to the possibility of occurrence of local minima in the potential field. The presence of local minima can lead to a stable positioning of the robot before it reaches its goal.

A second major non-configuration space approach was developed by Brooks [4, 6] and involves the use of generalized cones. These cones, used to represent the free space, are essentially straight lines, or "spines", with left and right free-space width functions. The allowable motions and robot orientations along the lines are found at each point along the spine. At the intersection points of the cones, the interval of safe orientation of the robot is calculated. A complete path is then found by transferring from one spine to another at their intersection, remaining within the safe robot orientation range at the intersections. Due to the fact that the cones may not cover all of the free space, this approach may fail to find a path when one actually exists. However, this method does have the appealing characteristic that the robot's path will stay away from obstacles.

D. T. Kuan, J. C. Zamiska, and R. A. Brooks [37, 38] build upon the generalized cone concept by using a mixed representation of free space. The generalized cones are used to represent narrow regions, or channels, while convex polygons represent large open spaces, due to the awkwardness of approximating large open spaces as cones. A connectivity graph is constructed in which the nodes represent the polygonal regions bounded by several obstacles, and the links represent the channels passing between two neighboring obstacles. An A* search is then used to find the optimal path, with a heuristic which estimates the remaining distance to the goal.

Many other non-configuration space approaches to path planning have been developed. J. T. Schwartz and M. Sharir developed a series of path planning methodologies dealing with several variants of the Piano Mover's problem [69, 70, 71, 72, 73]. These papers deal with the problems of finding paths for a rigid line segment, a rod, and a spider, and for coordinating the motion of a series of moving robots. W. E. Howden [27] examines the problem of determining whether an irregular two-dimensional object, or sofa, can be moved to a goal inside a complicated two-dimensional structure. C. Thorpe [81] uses a path relaxation method of finding safe robot movements, while E. G. Gilbert and D. W. Johnson [20] study the mathematical properties of distance functions which lead to the formulation of path planning problems as problems in optimal control.

It should be obvious that much work has been accomplished in the area of robot navigation for static environments. Although many of the algorithms perform quite well, no single approach to this problem is without its disadvantages. When selecting an algorithm for any specific static environment application, the advantages and disadvantages of these algorithms and the requirements of the application must be considered.

Coupled closely with these navigation algorithms are terrain acquisition algorithms which allow the robot to survey its environment to obtain an accurate world model. Once the world model is obtained, it is assumed to be static for planning purposes, and is subsequently used as input to one of the previously described navigation algorithms, such as the visibility graph method, to derive future path plans. R. S. Ahluwalia and E. Y. Hsu [2] and N. Rao, S. S. Iyengar, B. J. Oomen, and R. L. Kashyap [61] present algorithms using this method.

2.1.2 Heuristic Approaches. Heuristic approaches to robot navigation are often used when it is doubtful that a reliable world model can ever be obtained for path planning. These approaches rely primarily on sensor data to make intelligent decisions in the development of path plans. The majority of the sensor-based approaches utilize range data to detect and avoid obstacles in the environment. D. F. Cahn and S. R. Phillips [7], R. Chattergy [10], J. L. Crowley [13], Y. Ichikawa and N. Ozaki [29], D. M. Keirsey et al [34], E. Koch et al [35], V. Lumelsky and A. A. Stepanov [45], J. S. B. Mitchell [50], and C. R. Weisbin et al [83] have developed various range-based algorithms. Additional algorithms plan paths based upon vision feedback, such as the work described by A. Meystel and E. Koch [48] and H. P. Moravec [51].

The heuristic approaches typically operate effectively in practice, but cannot be proven to be correct in the sense of the algorithmic methods. Additionally, the resulting paths are not guaranteed to be optimal. However, the heuristic approaches may often find a good approximation to the optimal path in a much shorter period of time than an algorithmic method, and may thus be more appropriate for use in certain applications.

2.2 DYNAMIC ENVIRONMENT

One limitation shared by most of the algorithms described in the previous section is their difficulty with handling moving obstacles. The algorithms were designed for static robot environments and are not easily extended to dynamic environments. Admittedly, this limitation may not be serious for some applications, if the true world can accurately be modeled as static. Quite often, however, the real world will contain moving objects such as would be found on a highway or in a room inhabited by people. If the robot is to successfully navigate in these environments, it must be able to deal with moving obstacles.

Dynamic environments pose a more difficult path planning problem which has not yet been fully addressed. The most notable approach is the Avoid system developed by J. Lamadrid [39] which allows the robot to move through Cartesian space in the presence of objects moving in predictable trajectories. The algorithm operates in two dimensions and expands the objects, represented by their bounding circles, to allow the robot to be treated as a point. Both the robot and the obstacles have a time coordinate attached to them, and the obstacle trajectories are constrained to be represented as quadratic parametric equations in time.

To plan a path, Lamadrid's collision detection module finds the first collision point of the robot with any object along a straight-line trajectory from the starting point to the goal. When a collision is predicted, the replanner uses one of two methods for replanning the path at the collision point. The first method, the tangent method, involves the computation of a vector which is tangent to the colliding obstacle. According to a specified formula, the path is then directed along the tangent line to a subgoal located at a distance no greater than the radius of the colliding obstacle. The second replanning method, the field method, uses Khatib's artificial potential fields. The subgoal point is first computed using the tangent method, and then the artificial fields are used to move the computed subgoal to a more effective point. After computing this subgoal point, the algorithm then attempts to plan a straight-line path from the start to the subgoal, and then from the subgoal to the original goal. Although the Avoid system offers an interesting approach to the navigation problem, it will not typically produce optimal results, since the optimal path to the goal may not entail aiming directly at the goal at each planning step.

Of the algorithms described in the previous section, only Khatib's artificial potential field algorithm [33] seems to be readily adaptable to a dynamic environment. By allowing the artificial potential fields to continuously vary in time with the obstacle movement, the robot's planning algorithm is able to compensate for the object movement. The remainder of Khatib's approach would remain unchanged, and should, in theory, work equally well for moving obstacles as for static obstacles.

Unfortunately, extensions to the remainder of the previously described algorithms to allow the robot to deal with moving obstacles are difficult to achieve, although not impossible. To accomplish this, the algorithms must be expanded to include the expression of time, most likely as an added dimension to the world representation. Thus, rather than searching a two- (or three-) dimensional graph or tree structure, a three- (or four-) dimensional representation must be searched to find a valid path through time.

Recall that the visibility graph approach employed a network of straight line segments connecting each obstacle vertex with the remaining visible vertices. Extending this representation to moving obstacles results in curved surfaces replacing the straight-line segments between vertices, due to the movement of the obstacles relative to each other. To locate a path to the goal, a search of this collection of curved surfaces must then be performed — a difficult and complicated process. However, the algorithm developed in this thesis is similar to the visibility graph approach, in that the points in space at which the robot and the obstacles meet tangentially are searched in a manner analogous to the two-dimensional visibility graph method for static obstacles. This approach obviates the need to search the complex aggregation of curved surfaces for a path.

The extension to both the Voronoi diagram and the generalized cones approaches yields a representation resembling the extension to the visibility graph: a collection of curved surfaces. In this case, however, rather than representing the connections between the obstacle vertices, the curved surfaces represent paths at which the robot would remain as far as possible from the obstacles. Although this characteristic is appealing in a navigation algorithm for moving obstacles, it likewise requires the difficult search of the complex curved surfaces to find suitable robot paths.

Expanding the quadtree methodology to handle moving obstacles is particularly cumbersome, since the quadtree representation of free space does not localize the effect of obstacles. Thus, the entire tree representation of the world could potentially change due to a moving obstacle, rather than merely one or two branches of the tree. The tree representation would then be changing so often that the development of feasible paths would be difficult.

Other studies related to the dynamic environment path planning problem include work by R. M. Salter [67], J. W. Boyse [36], and S. Cameron [8]. These investigations, however, center around the detection of collisions between moving objects, without extending the research to the planning of paths based on the collision detections.

CHAPTER 3

DETAILS OF PATH PLANNING

3.1 REPRESENTATIONS AND EQUATIONS OF MOTION

3.1.1 Robot. The use of configuration space for navigation planning allows the robot to be viewed as a point such that any safe path derived for the point robot in configuration space will be safe for the actual robot in Cartesian space. Since the robot is operating in a dynamic world, this point representation must be extended to represent the passage of time. A desirable characteristic of the selected robot representation should be its ability to specify all valid locations of the robot in the two-dimensional x, y plane during the passage of time t. This characteristic is needed to ascertain all of the potential collision areas of the robot with the obstacles. For a constant robot speed, a cone representation of the robot possesses exactly this characteristic: it provides the two-dimensional coordinates of the possible robot locations at any value of time from a given starting location. Figure 5 illustrates the conical robot representation.

To derive the equation of the robot motion using this representation, first let (x_0, y_0) be the starting location of the robot at time $t = 0$, and let v be the constant velocity of the robot. For any given value of t, the possible robot locations form a circle with a general equation of:

$$(x - x_0)^2 + (y - y_0)^2 = (radius)^2$$

The radius of the circle equals the distance traveled by the robot in time t at constant velocity v from its starting location. Thus,

$$(radius)^2 = (vt)^2$$

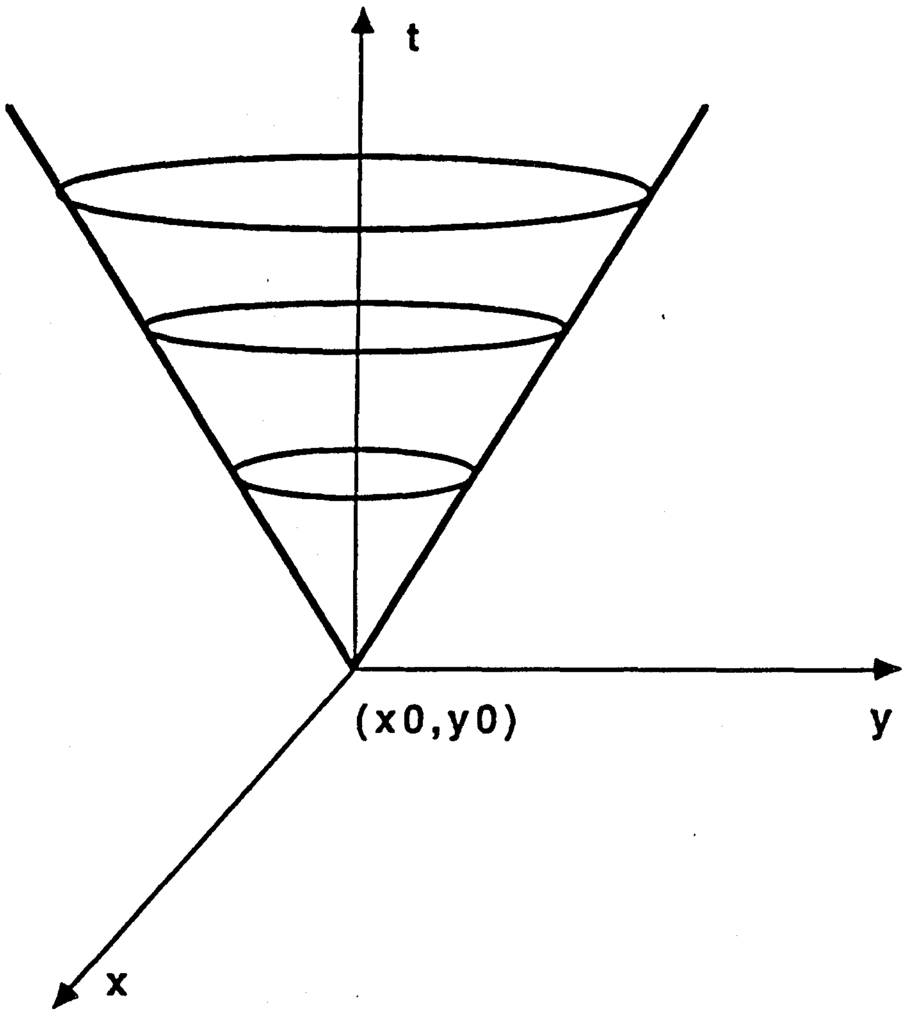


Figure 5. Robot representation as a cone.

and the equation of the robot motion, represented as a cone, is given by:

$$(x - x_0)^2 + (y - y_0)^2 = (vt)^2 \tag{1}$$

Of course, at any given instant in time during the execution of a path, the robot in actuality is no more than a simple point somewhere on the cone – only during the overall planning phase is the conical robot representation utilized.

3.1.2 Obstacles. Using the configuration space approach requires the obstacles to be expanded to correspond to all possible collisions between the obstacles and the point robot in two-dimensional Cartesian space. This thesis assumes that upon initiation of the path planning process, the obstacle growing operation has already been applied to the obstacles, approximated by their bounding circles, to convert the Cartesian space representation to the configuration space. An obstacle growing operation such as that described by Lozano-Perez and Wesley in [42] is useful for this purpose.

Once the obstacles have been grown, it is necessary to define those areas of configuration space which are swept out by each moving obstacle over time. Since the obstacle representation in the two-dimensional x, y plane is circular, the area swept out by an obstacle over time is in the shape of a sheared cylinder (see figure 6). To derive the equation of obstacle motion, let θ equal the direction of obstacle motion, k equal the constant velocity of the obstacle, (x_{start}, y_{start}) equal the starting location of the obstacle, and r equal the radius of the obstacle bounding circle in the configuration space. Next, note that the base of the sheared cylinder is a circle with the initial equation:

$$(x - x_{start})^2 + (y - y_{start})^2 = r^2 \tag{2}$$

As time passes, the radius of the sheared cylinder remains constant (since the obstacle is assumed to be rigid), but the location of the center of the circle

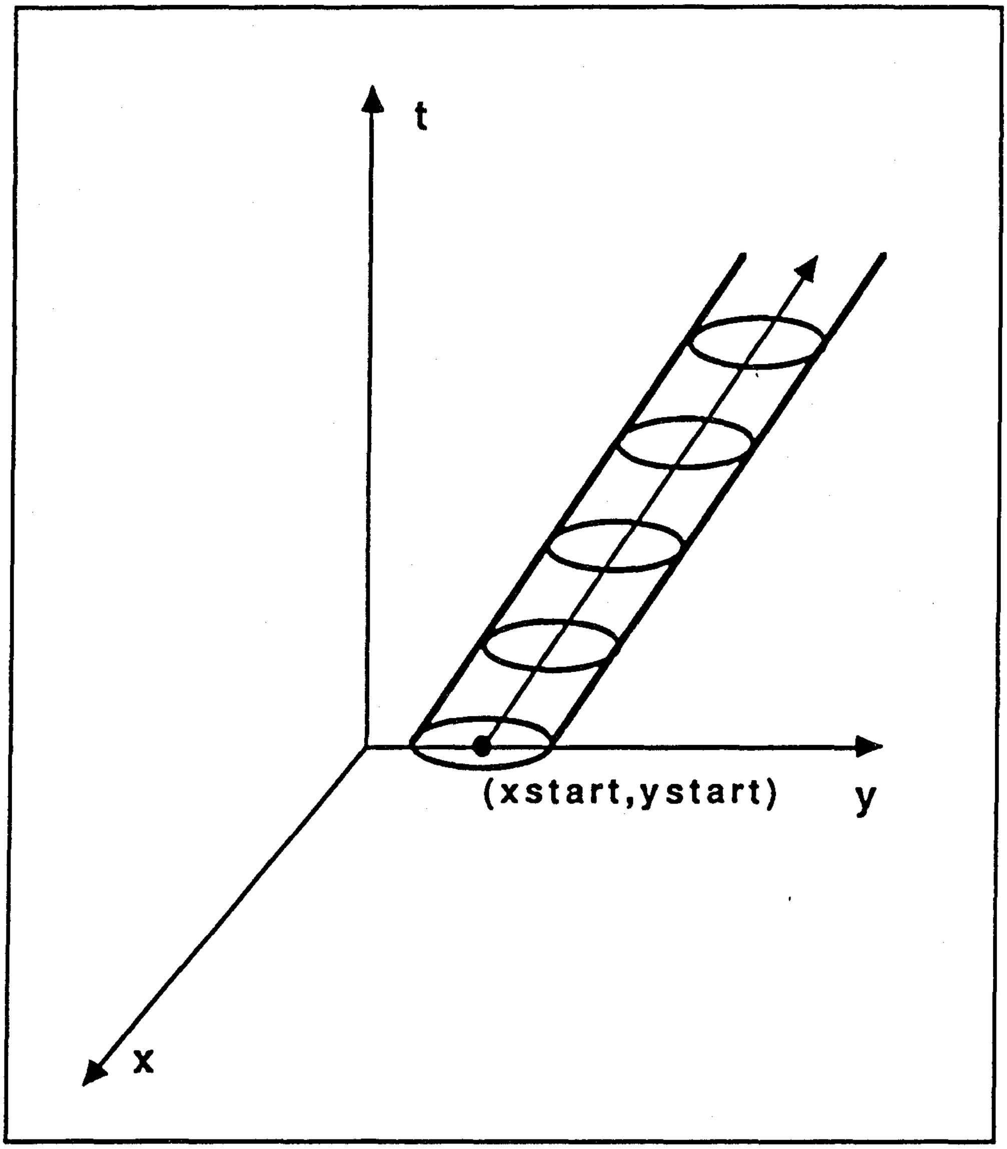


Figure 6. Obstacle representation as a sheared cylinder.

changes based upon the velocity of the obstacle, k, and its direction of movement, θ. To determine the center of the circle, (x_{next}, y_{next}), at time t, examine figure 7. In this figure,

$$\cos\theta = \frac{x_{next} - x_{start}}{kt}$$

$$\sin\theta = \frac{y_{next} - y_{start}}{kt}$$

Therefore,

$$x_{next} = kt\cos\theta + x_{start}$$

$$y_{next} = kt\sin\theta + y_{start}$$

Substituting these values into (2) yields:

$$(3) \qquad (x - x_{start} - kt\cos\theta)^2 + (y - y_{start} - kt\sin\theta)^2 = r^2$$

which is the general equation of motion of each of the obstacles. The specific motion equation for each obstacle is created by substituting the starting location, direction of movement, radius, and velocity associated with the individual obstacles into the above equation. Thus, at any time $t > 0$, the corresponding x, y values provide the boundary equation of the obstacle. In this thesis, the resulting obstacle representation implies collisions only when the robot is in the interior of the obstacle boundary; locations along the border of the obstacle representation are not collisions.

3.2 DERIVING INTERSECTION EQUATION

In order to determine the points at which the robot and the obstacles meet tangentially, it is first necessary to calculate all the points at which the robot could collide with each of the obstacles. The general intersection equation is derived by first solving for x in equations (1) and (3), the general equations of

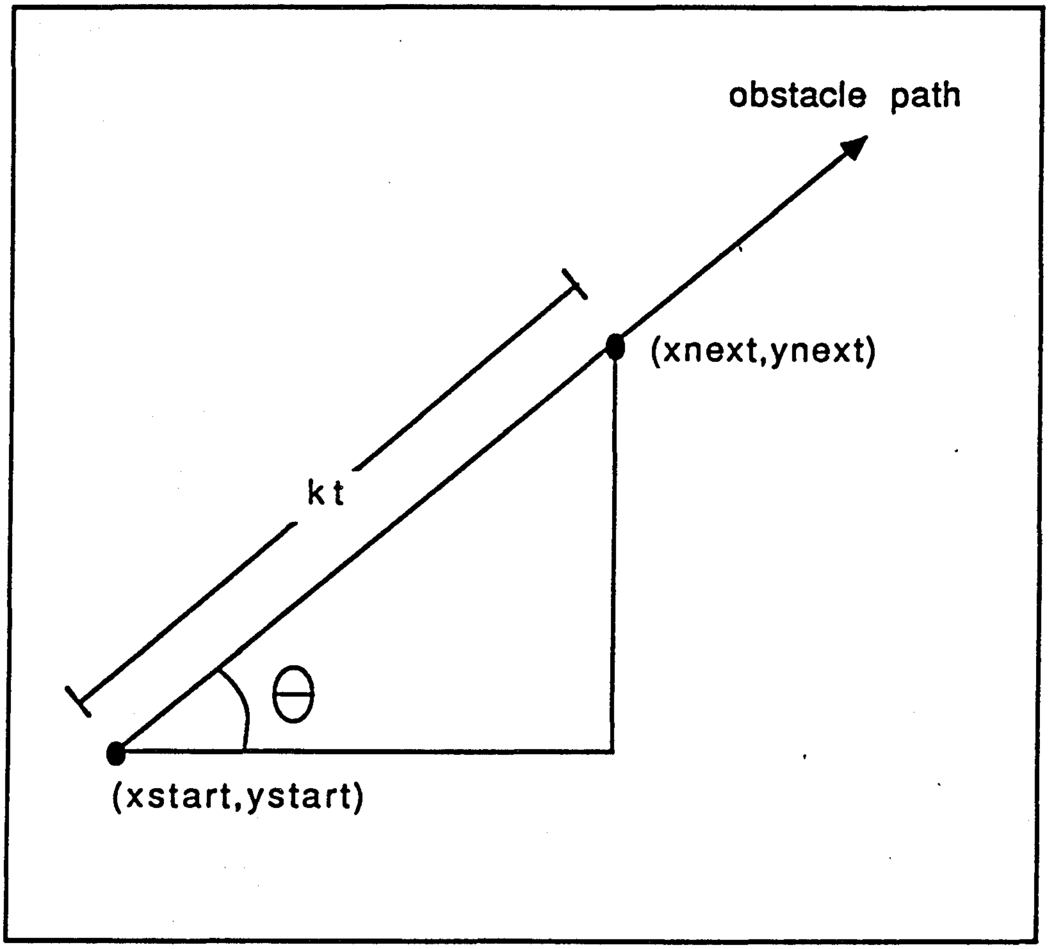


Figure 7. Geometry of motion of obstacle center.

robot and obstacle motion. Equation (1) then becomes:

$$x = x_0 \pm \sqrt{(vt)^2 - (y - y_0)^2} \tag{4}$$

and equation (3) becomes:

$$x = x_{start} + kt\cos\theta \pm \sqrt{r^2 - (y - y_{start} - kt\sin\theta)^2} \tag{5}$$

The intersection curve can then be determined by setting equations (4) and (5) equal to each other, and solving for y in terms of t. Setting the equations equal yields:

$$x_0 \pm \sqrt{(vt)^2 - (y - y_0)^2} = x_{start} + kt\cos\theta \pm \sqrt{r^2 - (y - y_{start} - kt\sin\theta)^2}$$

Significant algebraic manipulation is required to solve this equation for y in terms of t, eventually leading to the following intersection curve:

$$ay^2 + by + c = 0 \tag{6}$$

where:

$$\begin{aligned}
a = {} & 2(y_{start}y_0 + kty_0\sin\theta - kty_{start}\sin\theta) - y_{start}^2 \\
& - k^2t^2\sin^2\theta - (x_0 - x_{start} - kt\cos\theta)^2 - y_0^2 \\
b = {} & 2r^2y_0 - 2(-y_{start} - kt\sin\theta)(v^2t^2 - y_0^2) \\
& - 2y_0(-y_{start} - kt\sin\theta)^2 + \Big((x_0 - x_{start} - kt\cos\theta)^2 - r^2 \\
& - v^2t^2 + 2kty_{start}\sin\theta + k^2t^2\sin^2\theta + y_0^2 + y_{start}^2\Big) \\
& \times (y_{start} + kt\sin\theta + y_0) \\
c = {} & r^2v^2t^2 - y_0^2r^2 - (-y_{start} - kt\sin\theta)^2(v^2t^2 - y_0^2) \\
& - \frac{1}{4}\Big((x_0 - x_{start} + kt\cos\theta)^2 - r^2 - v^2t^2 \\
& + 2kty_{start}\sin\theta + k^2t^2\sin^2\theta + y_0^2 + y_{start}^2\Big)^2
\end{aligned}$$

Thus, the solution for y is given by:

$$y = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \tag{7}$$

This equation provides all possible x, y coordinates of intersection between the robot and the obstacles for selected values of t. The most effective choices for values of t are in an arithmetic progression $t_0, t_1, ..., t_n$ where $t_i = t_{i-1} + \delta t, t_0 = 0$, and $t_n = t_{max}$. (This approach has also been used by Cameron [8] to detect collisions.) The selection of δt is quite important, since solutions may be missed if δt is too large, or time could be wasted if δt is too small. Cameron [8] has experimented with the selection of δt, and has developed procedures useful for varying the value of δt based upon the distance between objects and their speeds. The approach taken by this thesis is to select a constant value of δt, which has proven effective in the implementation. The selection of t_{max} is somewhat application-dependent and determines how far into the future the intersection curves are detected. The value of t_{max} should be assigned to be at least equal to the time required for the robot to travel a straight-line path to the goal. Figure 8 gives two examples of the shape of the intersection equation projected onto the x, y plane.

Note that sections of the intersection curve correspond to intersections which can never occur in actuality. These portions of the curve reflect the impossible real-world situation in which the robot collides with an obstacle on one side, then travels through the interior of the obstacle to intersect the opposite boundary of the obstacle. However, these sections of the curve do not cause a problem to the planning methodology, which is able to plan successfully in spite of this situation.

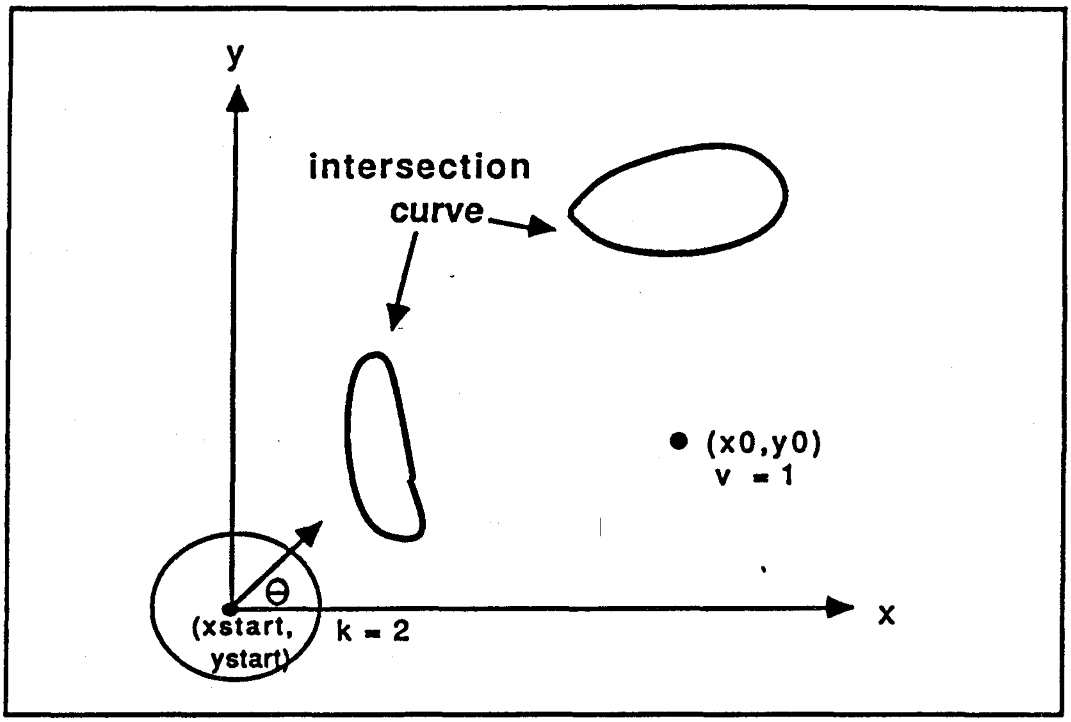


Figure a. Intersection curve with two continuous regions.

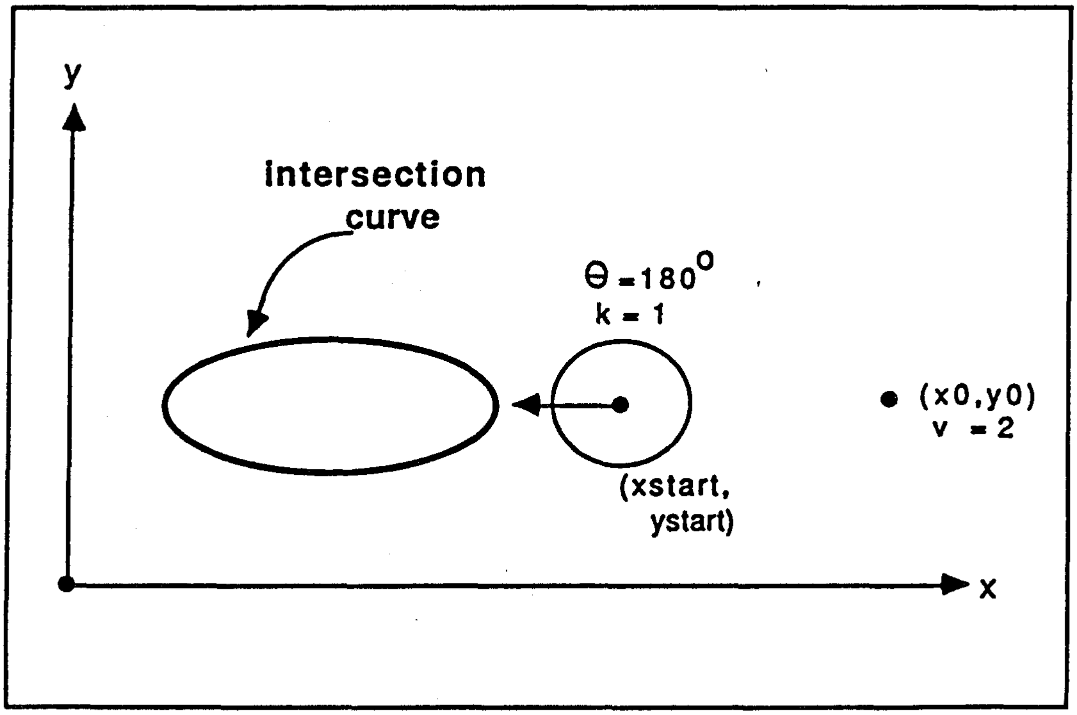


Figure b. Intersection curve with one continuous region.

Figure 8. Sample intersection curves.

3.3 COMPUTING THE DERIVATIVE OF THE INTERSECTION EQUATION

Once the intersection equation for each obstacle has been determined, the location of the tangent points must be computed. The tangent points are those locations in time at which the robot and the moving obstacles will meet tangentially. The first derivative of y with respect to x of the intersection equation is useful in calculating these tangent points. To compute the derivative, the intersection equation must be solved in terms of x and y, factoring out the t variables. First, the equation of motion of the robot, equation (1), must be solved for t, yielding:

$$t = \frac{\sqrt{(x-x_0)^2+(y-y_0)^2}}{v}$$

This equation can then be substituted into the intersection equation (6), resulting in:

$$\begin{aligned}
0 = y^2\Bigg(& 2y_{start}y_0 + \frac{2ky_0\sin\theta\sqrt{(x-x_0)^2+(y-y_0)^2}}{v} \\
& - \frac{2ky_{start}\sin\theta\sqrt{(x-x_0)^2+(y-y_0)^2}}{v} \\
& - y_{start}^2 - \frac{k^2\sin^2\theta((x-x_0)^2+(y-y_0)^2)}{v^2} \\
& - \left(x_0 - x_{start} - \frac{k\cos\theta\sqrt{(x-x_0)^2+(y-y_0)^2}}{v}\right)^2 - y_0^2\Bigg) \\
& + y\left(2r^2y_0 - 2y_0\left(-y_{start} - \frac{k\sin\theta\sqrt{(x-x_0)^2+(y-y_0)^2}}{v}\right)^2\right. \\
& + \left(y_{start} + \frac{k\sin\theta\sqrt{(x-x_0)^2+(y-y_0)^2}}{v} + y_0\right) \\
& \times \left(\left(x_0 - x_{start} - \frac{k\cos\theta\sqrt{(x-x_0)^2+(y-y_0)^2}}{v}\right)^2 - r^2\right. \\
& - ((x-x_0)^2+(y-y_0)^2) + \frac{2ky_{start}\sin\theta\sqrt{(x-x_0)^2+(y-y_0)^2}}{v}
\end{aligned} \tag{8}$$

$$+\frac{k^2\sin^2\theta((x-x_0)^2+(y-y_0)^2)}{v^2}+y_0^2+y_{start}^2\Bigg)$$

$$-2\left(-y_{start}-\frac{k\sin\theta\sqrt{(x-x_0)^2+(y-y_0)^2}}{v}\right)$$

$$\times((x-x_0)^2+(y-y_0)^2)$$

$$+2y_0^2\left(-y_{start}-\frac{k\sin\theta\sqrt{(x-x_0)^2+(y-y_0)^2}}{v}\right)\Bigg)$$

$$+r^2((x-x_0)^2+(y-y_0)^2)-y_0^2r^2$$

$$-\left(-y_{start}-\frac{k\sin\theta\sqrt{(x-x_0)^2+(y-y_0)^2}}{v}\right)^2$$

$$\times((x-x_0)^2+(y-y_0)^2)+y_0^2\Bigg(-y_{start}$$

$$-\frac{k\sin\theta\sqrt{(x-x_0)^2+(y-y_0)^2}}{v}\Bigg)^2$$

$$-\frac{1}{4}\Bigg(\left(x_0-x_{start}-\frac{k\cos\theta\sqrt{(x-x_0)^2+(y-y_0)^2)}}{v}\right)^2-r^2$$

$$-((x-x_0)^2+(y-y_0)^2)+\frac{2ky_{start}\sin\theta\sqrt{(x-x_0)^2+(y-y_0)^2}}{v}$$

$$+\frac{k^2\sin^2\theta((x-x_0)^2+(y-y_0)^2)}{v^2}+y_0^2+y_{start}^2\Bigg)^2$$

The derivative $\frac{dy}{dx}$ is typically determined by solving the equation for y in terms of x and computing the derivative of the right-hand side with respect to x. However, solving equation (8) for either x or y involves an overwhelming amount of tedious algebraic manipulation. Therefore, implicit differentiation is used to calculate the derivative of each term with respect to both x and y. After considerable simplification, the following equations result:

$$dxvariable=\frac{k(x-x_0)}{v\sqrt{(x-x_0)^2+(y-y_0)^2}}\Bigg(y\cdot eqn(-2y_0\cos\theta+2y\cos\theta$$

$$-2y_{start}\cos\theta+eqn\cdot\sin\theta)+\sin\theta\Bigg(((x-x_0)^2+(y-y_0)^2)$$

$$\times \left(-2y_{start} + y + \frac{yk^2\sin^2\theta}{v^2}\right) + yy_{start}^2 + 2yy_{start}y_0 + 2y_0^2 y_{start}$$

$$+ 2y^2y_0 - 2y^2y_{start} - 2yy_0^2 - 4yy_0y_{start} + 2yy_{start}^2 - r^2y + yy_0^2\Big)\Big)$$

$$+ 2\sqrt{(x-x_0)^2+(y-y_0)^2}\frac{(x-x_0)ky\sin\theta}{v}\left(1+\frac{k^2\sin^2\theta}{v^2}\right)$$

$$+ (x-x_0)\Bigg(\frac{2k^2\sin^2\theta}{v^2}(3yy_{start} + y_0y - 2y^2 - (x-x_0)^2)$$

$$+ 2\left(-\left(-y_{start} - \frac{k\sin\theta\sqrt{(x-x_0)^2+(y-y_0)^2}}{v}\right)^2 + yy_{start}\right.$$

$$\left. - \frac{k^2yeqn\cos\theta\sin\theta}{v^2} - yy_0 + r^2\right)\Bigg)$$

$$-\frac{1}{2}\Bigg(eqn^2 - r^2 + y_0^2 + y_{start}^2$$

$$+ ((x-x_0)^2 + (y-y_0)^2)\left(\frac{k^2\sin^2\theta}{v^2} - 1\right)$$

$$+ \frac{2ky_{start}\sin\theta\sqrt{(x-x_0)^2+(y-y_0)^2}}{v}\Bigg)$$

$$\times \Bigg(2(x-x_0)\Bigg(\frac{-keqn\cos\theta}{v\sqrt{(x-x_0)^2+(y-y_0)^2}}$$

$$-1 + \frac{ky_{start}\sin\theta}{v\sqrt{(x-x_0)^2+(y-y_0)^2}} + \frac{k^2\sin^2\theta}{v^2}\Bigg)\Bigg)$$

$$dyvariable = \frac{k(y-y_0)}{v\sqrt{(x-x_0)^2+(y-y_0)^2}}\Bigg(y\sin\theta\Bigg(((x-x_0)^2+(y-y_0)^2)$$

$$\left(1+\frac{k^2\sin^2\theta}{v^2}\right) + 2yy_0 - 2yy_{start} - y_0^2 - 2y_0y_{start} + 3y_{start}^2$$

$$+ eqn^2 - r^2\Bigg) + 2eqn\cos\theta(y^2 - yy_{start} - yy_0)$$

$$+ 2\sin\theta y_{start}(-((x-x_0)^2+(y-y_0)^2) + y_0^2)\Bigg)$$

$$+\frac{k\sin\theta\sqrt{(x-x_0)^2+(y-y_0)^2}}{v}$$

$$\times\left(2y\left(-2y_{start}+2y+(y-y_0)\left(-1+\frac{k^2\sin^2\theta}{v^2}\right)\right)\right.$$

$$+((x-x_0)^2+(y-y_0)^2)\left(1+\frac{k^2\sin^2\theta}{v^2}\right)$$

$$\left.-y_0^2+3y_{start}^2+eqn^2-r^2+2y_0y_{start}\right)$$

$$+((x-x_0)^2+(y-y_0)^2)\left(\frac{k^2\sin^2\theta}{v^2}(3y_{start}-2y+y_0)\right.$$

$$\left.-y_{start}-y_0\right)+eqn^2(-2y+y_{start}+y_0)+\frac{2k^2\sin^2\theta(y-y_0)}{v^2}$$

$$\times\left(-yy_0+3yy_{start}-y^2+y_0^2-((x-x_0)^2+(y-y_0)^2)\right)$$

$$+y_{start}\left(-2yy_{start}+2y^2+2((x-x_0)^2+(y-y_0)^2)-y_0^2\right.$$

$$\left.-r^2+2y_0y+y_{start}^2+y_0y_{start}\right)+y_0\left(r^2\right.$$

$$-2\left(y_{start}+\frac{k\sin\theta\sqrt{(x-x_0)^2+(y-y_0)^2}}{v}\right)^2$$

$$\left.-2y^2+y_0^2\right)+(y-y_0)\left(\frac{-2k^2yeqn\sin\theta\cos\theta}{v^2}+2r^2\right.$$

$$\left.-2\left(y_{start}+\frac{k\sin\theta\sqrt{(x-x_0)^2+(y-y_0)^2}}{v}\right)^2\right)$$

$$-\frac{1}{2}\left(eqn^2-r^2+((x-x_0)^2+(y-y_0)^2)\left(-1+\frac{k^2\sin^2\theta}{v^2}\right)\right.$$

$$\left.+\frac{2ky_{start}\sin\theta\sqrt{(x-x_0)^2+(y-y_0)^2}}{v}+y_0^2+y_{start}^2\right)$$

$$\times\left(\frac{-2keqn\cos\theta(y-y_0)}{v\sqrt{(x-x_0)^2+(y-y_0)^2}}\right.$$

$$-2(y-y_0)+\frac{2ky_{start}\sin\theta(y-y_0)}{v\sqrt{(x-x_0)^2+(y-y_0)^2}}$$

$$\left.+\frac{2k^2\sin^2\theta(y-y_0)}{v^2}\right)$$

where

$$eqn = x_0 - x_{start} - \frac{k \cos\theta \sqrt{(x-x_0)^2 + (y-y_0)^2}}{v}$$

The derivative $\frac{dy}{dx}$ is then computed as follows:

$$\frac{dy}{dx} = -\frac{dxvariable}{dyvariable} \tag{9}$$

This equation provides the derivative at any point (x, y) which is a valid solution to the intersection equation (7). The following section discusses the usefulness of this derivative information.

3.4 FINDING TANGENT POINTS

Using the derivative of the intersection equation to calculate the tangent points is very straightforward after one observation: at a valid tangent point, the value of the derivative, $\frac{dy}{dx}$, will be the same as the slope of the line connecting the robot's current location and the intersection point. An analysis of the intersection curve will show why this observation must be true.

Consider the rays which emanate from the current location of the robot. These rays will intersect the intersection curve at either zero, one, or two points. Two crossings indicate that the robot would (if physically possible) collide with one side of the obstacle, travel through the obstacle, and leave through the back side of the obstacle. On the other hand, if the robot were to travel along a ray which did not intersect the intersection curve, it would not meet the obstacle at all. Finally, if the robot were to travel along a ray which intersected the intersection curve exactly once, it must skim the obstacle tangentially. At such a tangent point, the relative velocities of the robot and the obstacle will be such that the robot and the obstacle will travel apart. If this were not true, the robot would be trying to enter the obstacle and there would then be two distinct intersection points.

The tangential rays will intersect the intersection curve at the points at which the value of the derivative is the same as that of the intersecting ray. Thus, as each intersection point is computed, the derivative at that point can be calculated. The point can be classified as a tangent if it is within a small threshold of the slope of the intersecting ray. Notice that an exact match of slopes is unlikely due to the selection of the evaluation values of t. However, the match is close enough to give excellent experimental results, as described in Chapter 5. Figure 9 depicts an example of the intersection curve, the computed slopes, and the selected tangent points.

3.5 MODIFYING TANGENTS TO AVOID COLLISIONS

An analysis of the geometry of the robot and obstacle relationships shows that when a selected tangent point lies on the leading edge of an obstacle which is moving faster than the robot, the only subsequent option for the robot after reaching the tangent is to continue along its path until it has cleared the obstacle. If the robot tried to turn in towards the obstacle, it would immediately collide with the obstacle, while a turn away from the obstacle would result in a non-optimal path. Therefore, to save processing time, the tangent is extended to the point at which the obstacle is cleared. The following sections discuss how this extension is calculated.

3.5.1 Detecting Whether a Tangent is on a Leading or a Trailing Edge. Determining whether a tangent is on a leading or a trailing edge involves a geometrical analysis of the tangential meeting between the robot and the obstacle. Figure 10 shows the obstacle location at time t — the time at which the robot reaches the tangent point, V, or (x_v, y_v). As before, (x_{start}, y_{start}) is the starting position of the obstacle, k is the velocity of the obstacle, and θ is the

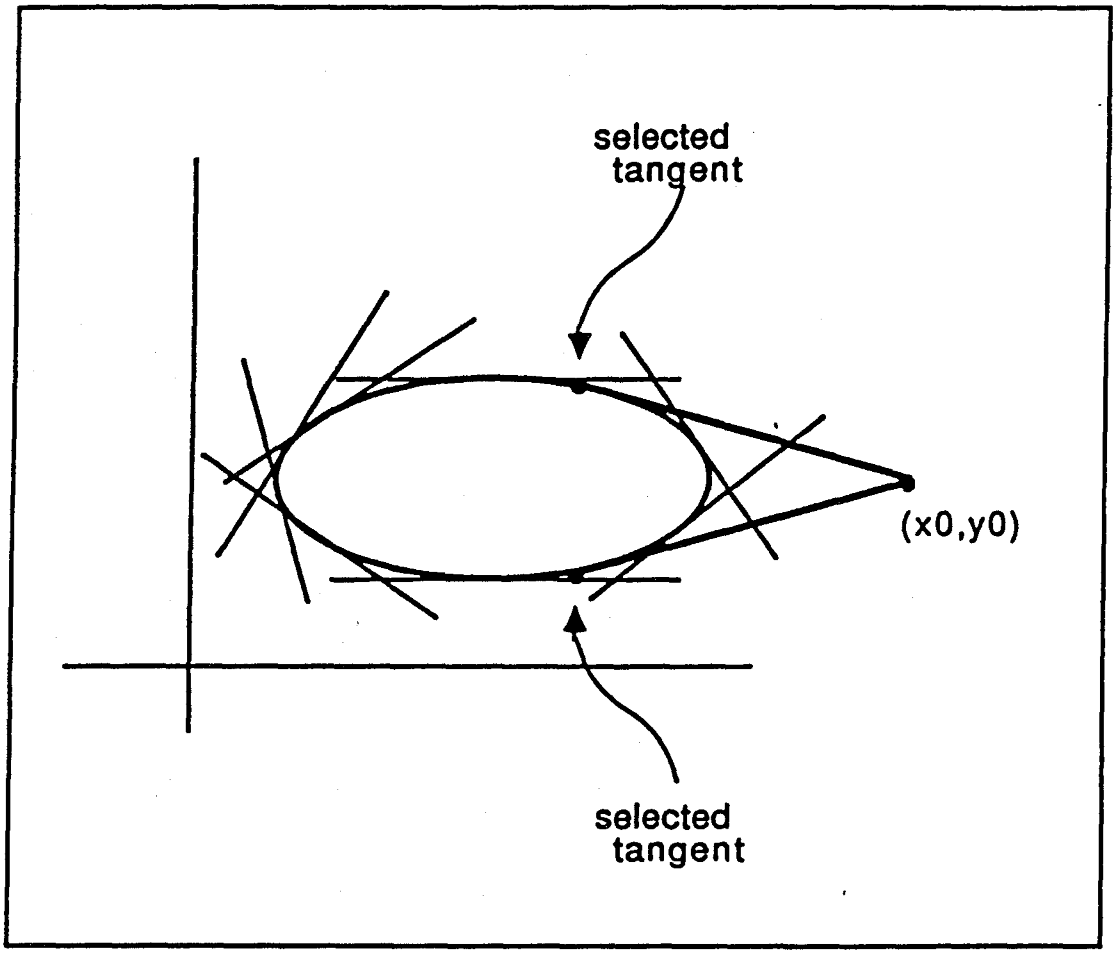


Figure 9. Intersection curve with slopes and selected tangent points.

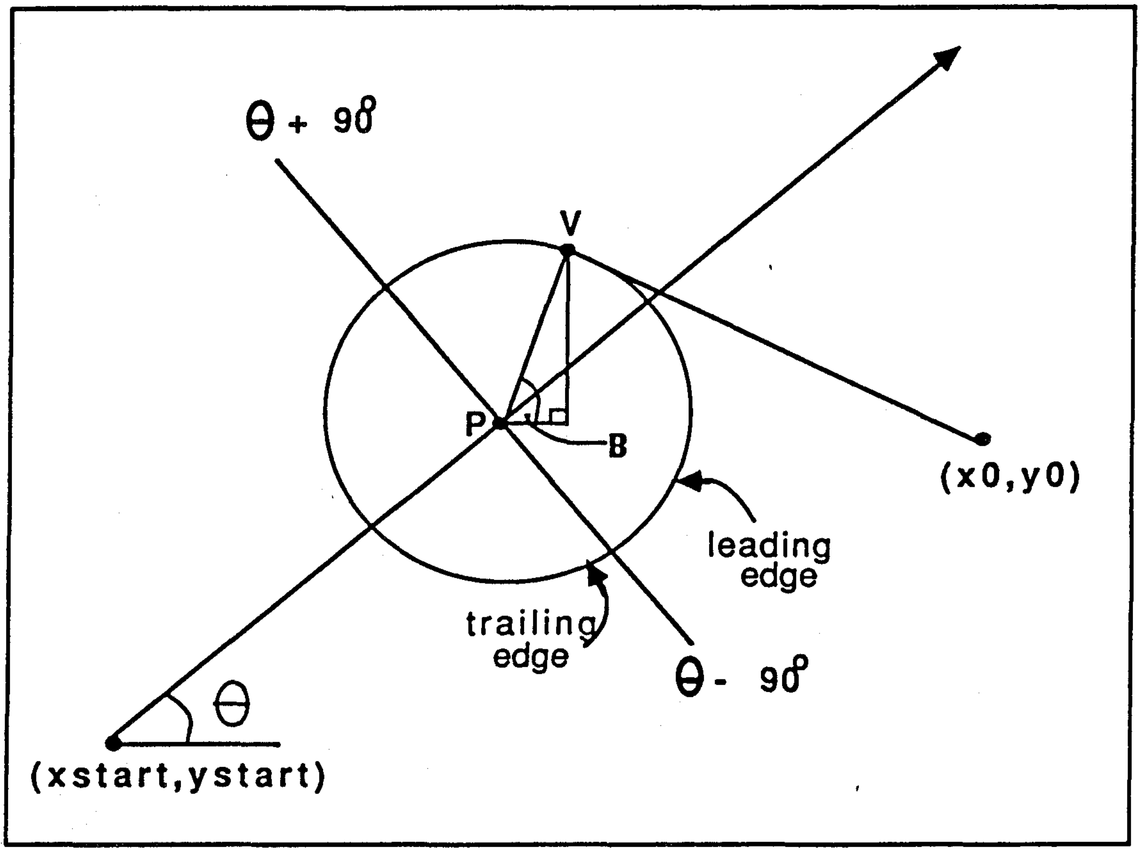


Figure 10. Geometry of tangential meeting of robot and obstacle.

direction of obstacle travel. The center of the obstacle at time t is given by:

$$(x_p, y_p) = (x_{start} + kt\cos\theta, y_{start} + kt\sin\theta)$$

The angle β from the current obstacle center to the tangent point is given by:

$$\beta = \tan^{-1}\frac{y_v - y_p}{x_v - x_p}$$

or,

$$\beta = \tan^{-1}\frac{y_v - y_{start} - kt\sin\theta}{x_v - x_{start} - kt\cos\theta}$$

The tangent point (x_v, y_v), then, is on the leading edge if the angle β lies in the range:

$$[\theta - 90^o, \theta + 90^o]$$

Thus, detecting whether a tangent is on a leading or a trailing edge requires computation of the angle β, followed by a simple comparison to the angle range above to determine the tangent's location on the obstacle boundary.

3.5.2 Extending a Tangent.

The extended tangent point must reach a distance r, the obstacle's radius, from the path of obstacle motion in order to clear the object. Figure 11 shows the initially selected tangent V and the desired extended tangent S. The first step towards calculating point S is to compute the intersection point, I, between the line of obstacle movement, l, and the line of robot movement, p. The equation of line p can be derived as:

$$y = \frac{y_v - y_0}{x_v - x_0}(x - x_0) + y_0$$

The equation of the line l is given by:

$$y = \tan\theta(x - x_{start}) + y_{start}$$

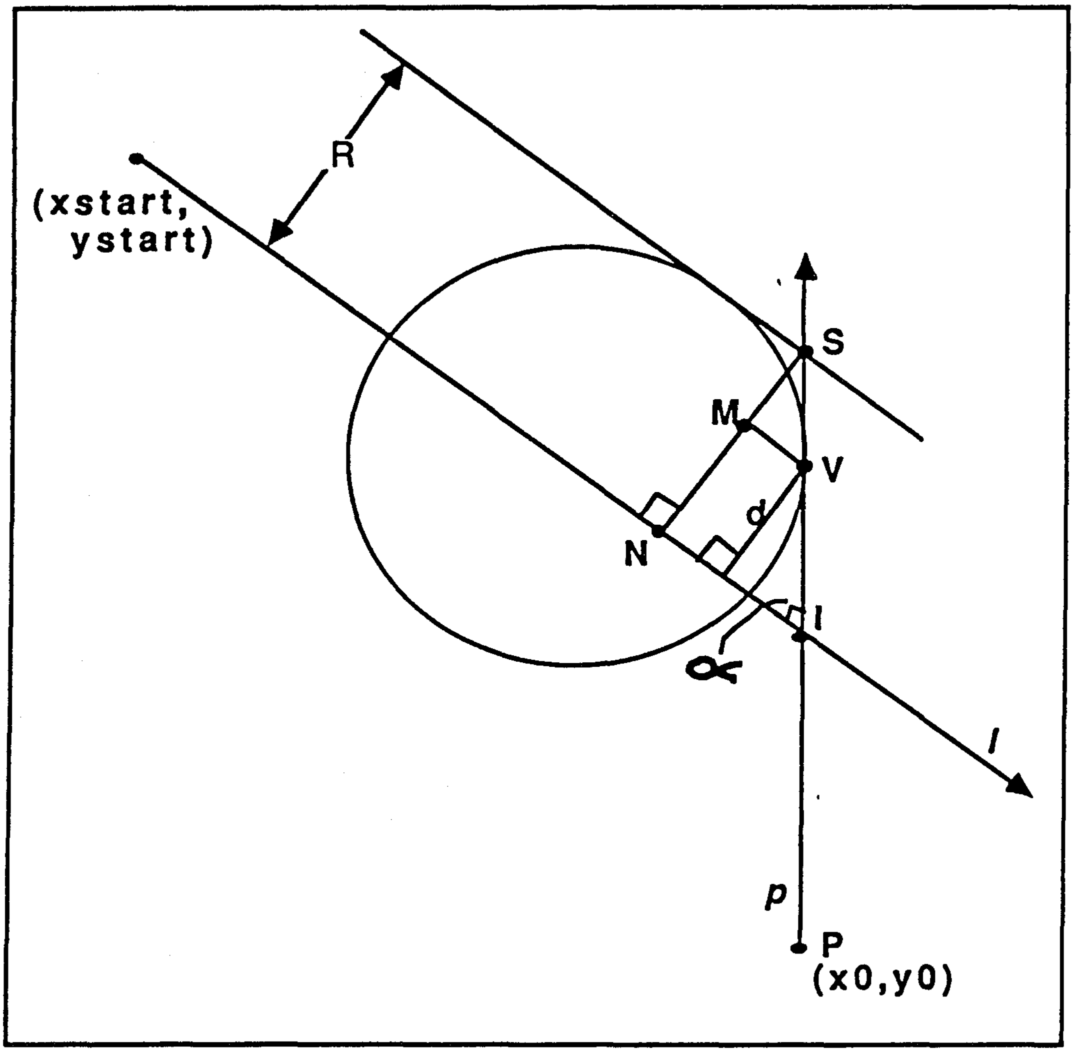


Figure 11. Geometry for extension of tangent.

while the intersection of the two lines at (x_I, y_I) can then be computed to be:

$$x_I = \frac{x_{start}\tan\theta - x_0(\frac{y_v - y_0}{x_v - x_0}) + y_0 - y_{start}}{\tan\theta - \frac{y_v - y_0}{x_v - x_0}}$$

$$y_I = \tan\theta(x_I - x_{start}) + y_{start}$$

Once the intersection point has been derived, the current distance, d, of the tangent point from the line of obstacle travel can be calculated as:

$$d = \sin\alpha\sqrt{(x_v - x_I)^2 + (y_v - y_I)^2}$$

where α is the acute angle at which lines p and l meet. As can be seen in figure 11, the extended tangent, S, will be the point at which the distance from N to S equals the radius of the obstacle, and the distance from M to S equals the radius minus the value of d. Thus, the extended tangent (x_s, y_s) will cause both of the following equations to hold true:

$$\sin\alpha\sqrt{(x_s - x_v)^2 + (y_s - y_v)^2} = r - d$$

$$\sin\alpha\sqrt{(x_s - x_I)^2 + (y_s - y_I)^2} = r$$

The solution to this system of equations is given by:

$$x_s = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

$$y_s = \frac{y_v - y_I}{x_v - x_I}(x_s - x_I) + y_I$$

where

$$a = 1 + \frac{(y_v - y_I)^2}{(x_v - x_I)^2}$$

$$b = -2x_v - 2x_I\frac{(y_v - y_I)^2}{(x_v - x_I)^2} + 2(y_I - y_v)\frac{y_v - y_I}{x_v - x_I}$$

$$c = x_v^2 + x_I^2\frac{(y_v - y_I)^2}{(x_v - x_I)^2} - 2x_I(y_I - y_v)\frac{y_v - y_I}{x_v - x_I} + (y_I - y_v)^2 - \frac{(r - d)^2}{\sin^2\alpha}$$

The originally computed tangent point V is thus replaced by the extended tangent point S, or (x_s, y_s) to allow the robot to clear an obstacle moving faster than the robot.

3.6 ANALYZING THE VISIBILITY OF THE TANGENT POINTS

Once the tangent points for each obstacle have been calculated, it is necessary to determine the visibility of each of the tangent points; i.e., whether each tangent point can be reached by the robot without colliding with another obstacle on route to the tangent point, and thus visible, or whether the tangent is unreachable by the robot, and thus hidden. An examination of the intersection curves will provide insight on how the visibility analysis can be successfully accomplished. As can be seen in figure 9, the function depicting the intersection points of the robot and an obstacle is characterized by continuous regions of collision and regions which have no solution, corresponding to those positions at which the robot and the obstacle will not collide. In each of the continuous regions, due to the circular obstacle representation in the x, y plane, at most two tangents will exist — one on either side of the obstacle. The region swept out behind the two tangents is unreachable by the robot. For each obstacle, there may be up to two continuous regions of the intersection curve, for a total of four possible tangent points per obstacle. The four tangent points correspond to the physical situation in which, due to a difference in robot and obstacle speeds, the robot can tangentially meet both the leading and the trailing edges of an obstacle twice. In figure 12, points P, Q, R, and S are the tangents of the intersection curve resulting from one obstacle, and points T, U, V, and W are the tangents of the intersection curve resulting from a second obstacle. Note that tangents T and W are hidden from the robot, because they are in the regions swept out by lines AP and AQ, and lines AR and AS, respectively. Thus, any tangent which would require the robot to travel a path crossing the intersection curve, or through another obstacle, must be marked as hidden. Of course, this situation must only be considered when more than one obstacle is present in the environment.

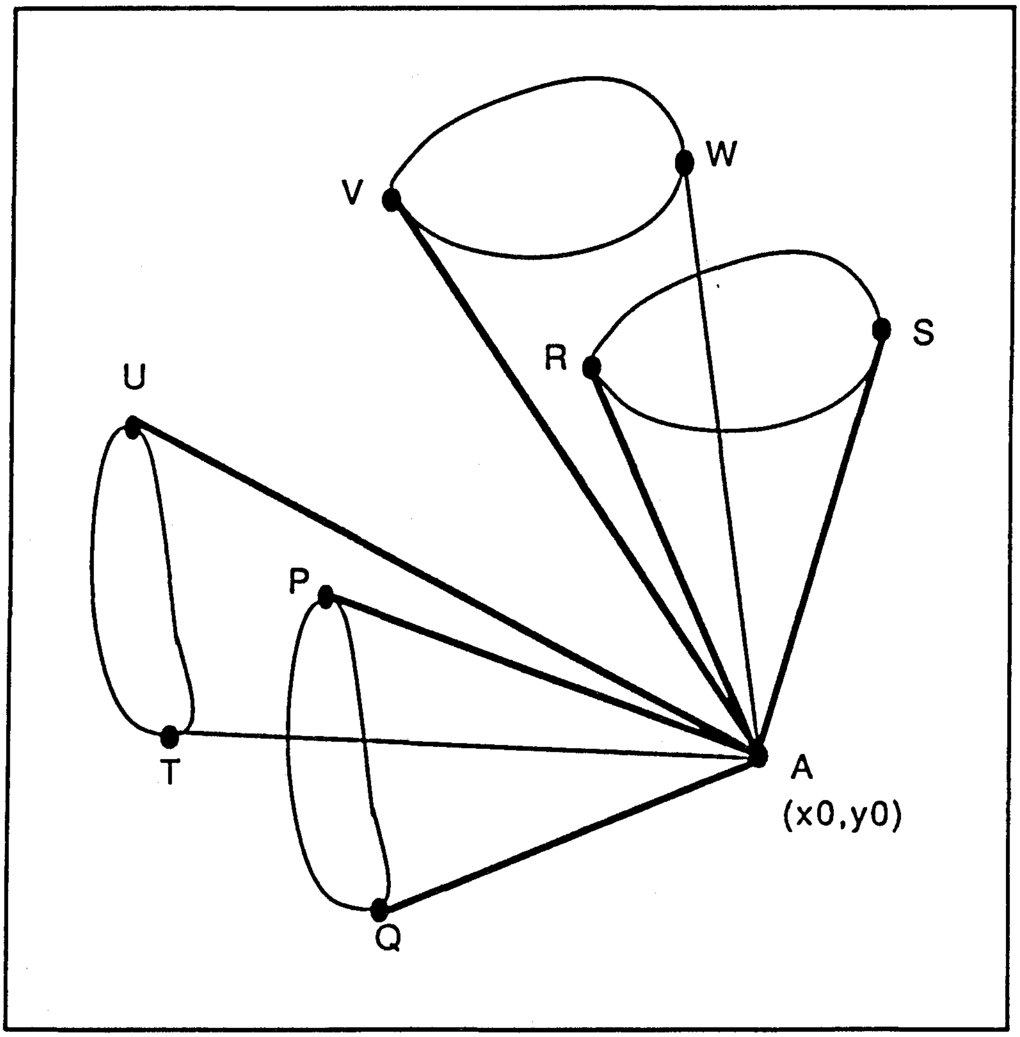


Figure 12. Sample intersection curve with hidden and visible tangents.

The exact location of the robot at any instant of time while it moves toward a tangent (x_v, y_v) from its starting location (x_0, y_0) is given by:

(10) $$x = x_0 + vt\cos\alpha$$

(11) $$y = y_0 + vt\sin\alpha$$

where α is the direction of robot travel to the tangent and is given by:

$$\alpha = \tan^{-1}\frac{y_v - y_0}{x_v - x_0}$$

Recall the equation of obstacle motion, repeated here for convenience:

(3) $$(x - x_{start} - kt\cos\theta)^2 + (y - y_{start} - kt\sin\theta)^2 = r^2$$

To find the possible collisions of the robot and an obstacle along the path to a tangent point which is associated with a different obstacle, substitute equations (10) and (11) into (3) to result in:

$$(x_0 + vt\cos\alpha - x_{start} - kt\cos\theta)^2 + (y_0 + vt\sin\alpha - y_{start} - kt\sin\theta)^2 = r^2$$

Solving for t in this equation gives:

(12) $$t = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

where:

$$a = v^2\cos^2\alpha - 2kv\cos\alpha\cos\theta + k^2\cos^2\theta$$
$$+ v^2\sin^2\alpha - 2kv\sin\alpha\sin\theta + k^2\sin^2\theta$$

$$b = 2vx_0\cos\alpha - 2x_0k\cos\theta - 2x_{start}v\cos\alpha + 2x_{start}k\cos\theta$$
$$+ 2vy_0\sin\alpha - 2y_0k\sin\theta - 2y_{start}v\sin\alpha + 2y_{start}k\sin\theta$$

$$c = x_0^2 - 2x_0x_{start} + x_{start}^2 + y_0^2 - 2y_0y_{start} + y_{start}^2 - r^2$$

The real-valued solutions for t, if they exist, provide the times at which the robot would collide with this obstacle along its path to the tangent point. The time required for the robot to reach the tangent is equal to:

$$\frac{\sqrt{(x_v - x_0)^2 + (y_v - y_0)^2}}{v}$$

Thus, if the solutions for t in equation (12) lie in the time interval:

$$\left[0, \frac{\sqrt{(x_v - x_0)^2 + (y_v - y_0)^2}}{v}\right]$$

then the path to the tangent point will result in a collision of the robot with this obstacle, and the tangent point should therefore be marked as hidden. This process should be repeated for each obstacle in the environment to finalize the decision on the visibility of the tangent. Only after failing to find collisions with any obstacle should the tangent be declared visible.

3.7 ANALYZING THE VISIBILITY OF THE GOAL

Detecting whether the goal is visible or hidden is analogous to determining the visibility of a tangent point, as described in the previous section. The only variation is in the range of time for which solutions of t imply a collision. The range of time in this case is given by:

$$\left[0, \frac{\sqrt{(x_{goal} - x_0)^2 + (y_{goal} - y_0)^2}}{v}\right]$$

where the goal is given by (x_{goal}, y_{goal}). Thus, each obstacle must be examined to determine if it causes a collision with the robot along the path to the goal. If no obstacle produces such a collision, the goal can be marked as visible and is reachable from the current tangent point.

CHAPTER 4

PATH PLANNING ALGORITHM

The path planning algorithm developed in this thesis is analogous to the visibility graph method for two-dimensional static environments, using an A* search to find the path to the goal. After computing the intersection equations and using the derivative information to find tangent points from the robot's current location, new paths are formed to each of the visible tangents and are assigned a cost. This cost is the sum of the elapsed time along the path to the tangent, plus a heuristic estimate of the amount of time remaining until the robot reaches the goal. The estimate of remaining time is equal to the time that would be required for the robot to travel a straight-line path to the goal from the current tangent point. The list of possible paths is stored in ascending order by cost, and the next tangent to be expanded is the last node on the path with the least cost. The process of expanding least-cost tangents is repeated until the robot reaches the goal, or until a maximum planning time is exceeded.

Given the starting location of the robot, (x_0, y_0), the goal to be reached, (x_{goal}, y_{goal}), the speed of the robot, v, the set $O = \{ O_1, ..., O_{nobst} \}$ giving data on the starting location, speed, radius, and direction of motion of each of the obstacles, and the maximum planning time (max-plan-time), the details of the path planning algorithm are as follows:

I. Create an initial path from the robot's starting location (x_0, y_0) to nowhere, having an initial cost of zero. This path becomes the only path in S, the set of potential paths to the goal.

II. While the last tangent, TZ, of the first path in S, $S1$, is not equal to the goal, and the elapsed time of the path $S1$ is less than max-plan-time, do the following:

  A. Determine if the goal is reachable from the last tangent of the first path in S. If so:

    1. Modify $S1$ by concatentating the goal point to the end of the path.

    2. Compute the cost as the elapsed time to reach the goal.

    3. Insert the modified path into S in ascending order according to this cost.

  B. Else (i.e. goal not visible)

    1. Expand the last tangent, TZ, of the first path in S to find the new set of tangents $T = \{ T_1, T_2, ..., T_n \}$ which are reachable from TZ. This is achieved as follows:

      a. Initialize the set T to the empty set $\{\}$.

      b. For each obstacle O_i in $O, i = 1, ..., nobst$:

        (1) Compute the intersection curve of the obstacle's sheared cylinder and the robot's conical representation.

        (2) Compute the derivative $\frac{dy}{dx}$ of the intersection curve.

        (3) Find the set of tangent points $U_i = \{ U_{i1}, U_{i2}, ... \}$ along the intersection curve.

        (4) Extend the tangent points of U_i by doing the following for each U_{ij} in U which corresponds to an obstacle moving faster than the robot:

          (a) Determine if U_{ij} is on the leading or trailing edge of the obstacle.

```
            (b) If $U_{ij}$ is on the leading edge, do the following: extend the
                tangent $U_{ij}$ to the point at which it clears the obstacle;
                replace $U_{ij}$ with the newly extended tangent.
        c. Determine the visibility of the tangent points in the $U_{ij}$ sets, $i =
           1, ..., nobst$ found in step II.B.1.b. Add all visible tangents $U_{ij}$ to
           $T$, forgetting all hidden tangents.
    2. Remove $S1$ from $S$, saving a copy. Then, for each tangent $T_k$ in $T$:
        a. Create a new path by concatenating $T_k$ to the end of the copy of
           $S1$.
        b. Compute the cost of the new path by adding the elapsed time to
           reach tangent $T_k$ and the estimated time to reach the goal, from
           $T_k$.
        c. Insert the new path into $S$ in ascending order by cost.
III. If the last tangent of $S1$ equals the goal, announce success;
    Else, announce failure
```

The complexity of the node expansion used by this algorithm is related to the number of tangent points found from the current robot location. For each obstacle, the maximum number of tangent points reachable from the current location is four, as illustrated in figure 8a on page 29. Thus, the maximum total number of tangents which can be found from the current robot location is $4 \times nobst$, where *nobst* equals the number of obstacles in the environment. Thus, the complexity of the node expansion is O(*nobst*). It is interesting to note that the maximum number of tangent points per stationary obstacle is two rather than four. Thus, the moving obstacle problem can be twice as complex as the stationary obstacle problem, although both are O(*nobst*) for a node expansion.

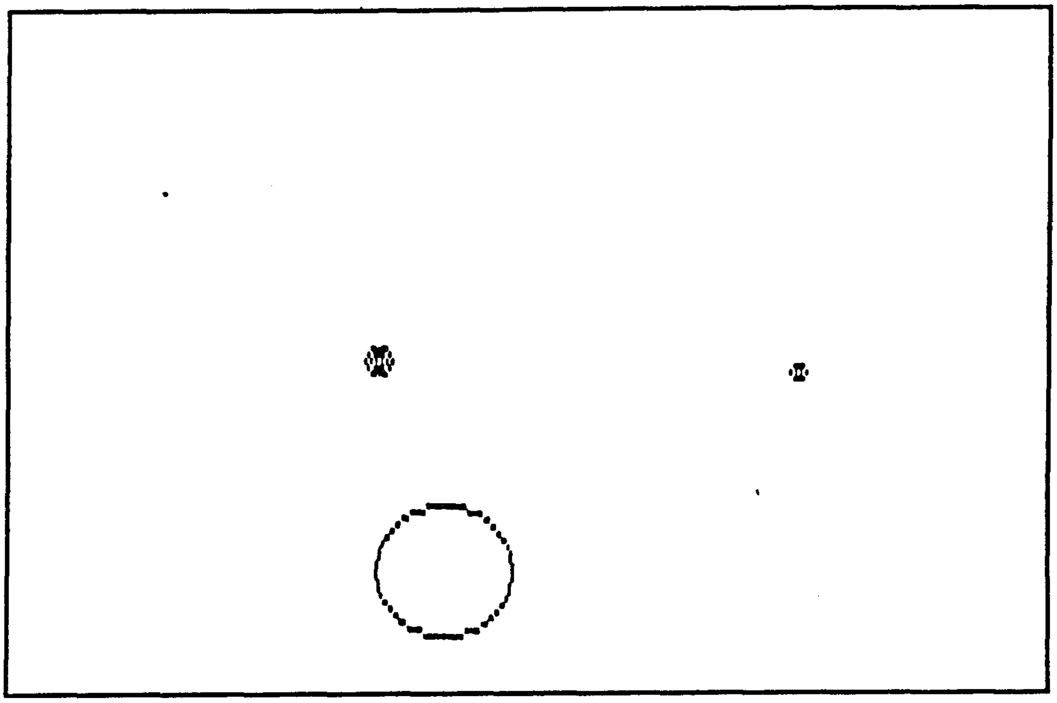
CHAPTER 5

RESULTS

The robot navigation algorithm described in this thesis has been implemented in the C programming language and has demonstrated the ability to successfully plan paths for a robot amidst moving obstacles. Figures 13 through 17 give computer output of several example paths planned by the algorithm. These figures are actually a series of snapshots of the robot executing its path while the obstacles move. Note that in these pictures, the endpoint of the line depicting the robot path is the actual location of the robot. Intersections of other portions of the line with an obstacle are not a problem, since they merely indicate that an obstacle is traveling in an area where the robot has already passed. (In these diagrams, the jagged edges of the obstacles and the robot path are due to the resolution of the computer graphics equipment rather than the navigation algorithm.)

Figures 13 and 14 illustrate the robot curving around obstacles, while figure 15 demonstrates the algorithm's ability to detect when no collisions will occur on the path to the goal. Figure 16 illustrates a robot path with sharp corners — a situation which will occur when the robot has cleared an obstacle moving faster than the robot. An actual real-world mobile robot may or may not be able to execute a path with such sharp turns, depending upon the capabilities of the specific robot. If not, some type of path-smoothing algorithm must be included to result in a path which can be successfully executed by the robot.

Figure 17 illustrates a situation in which the ability to vary the robot speed would be useful. In this example, the robot is moving faster than the obstacle, which travels directly over the goal point. To reach the goal, the robot



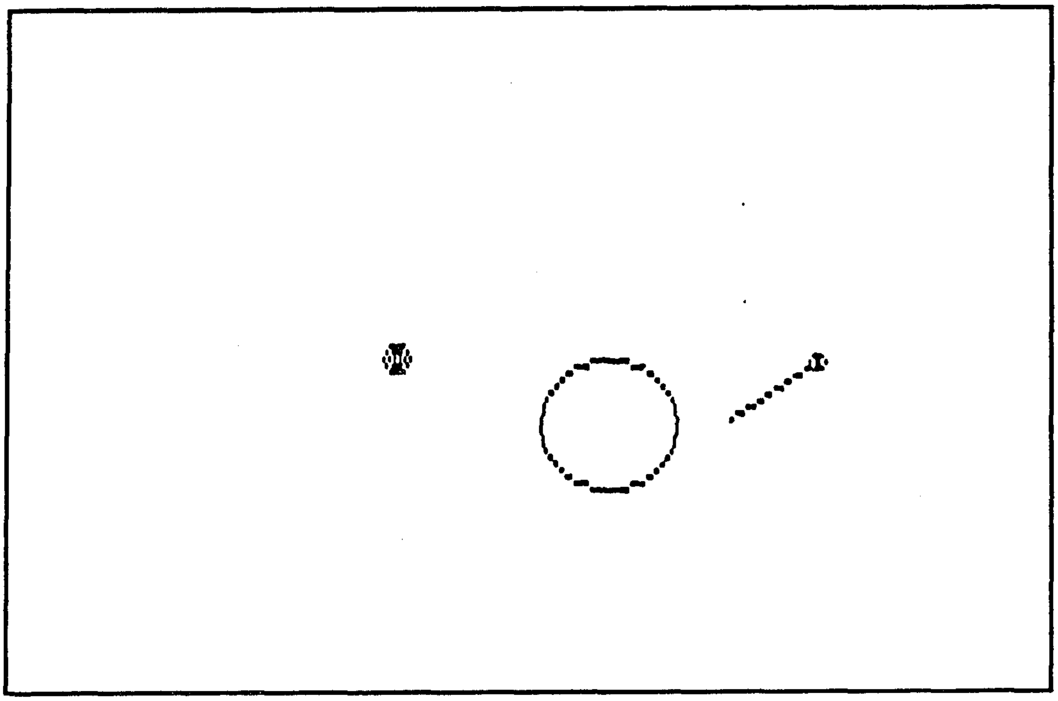


Figure 13. Example one: Robot path with one obstacle.

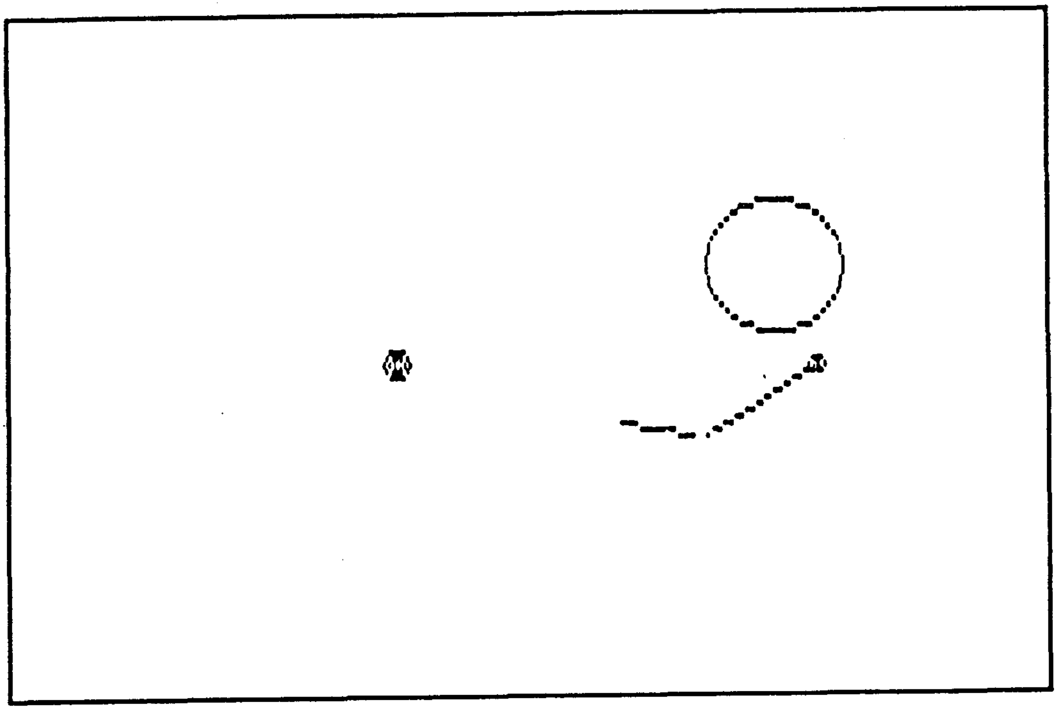


Figure 13, continued.

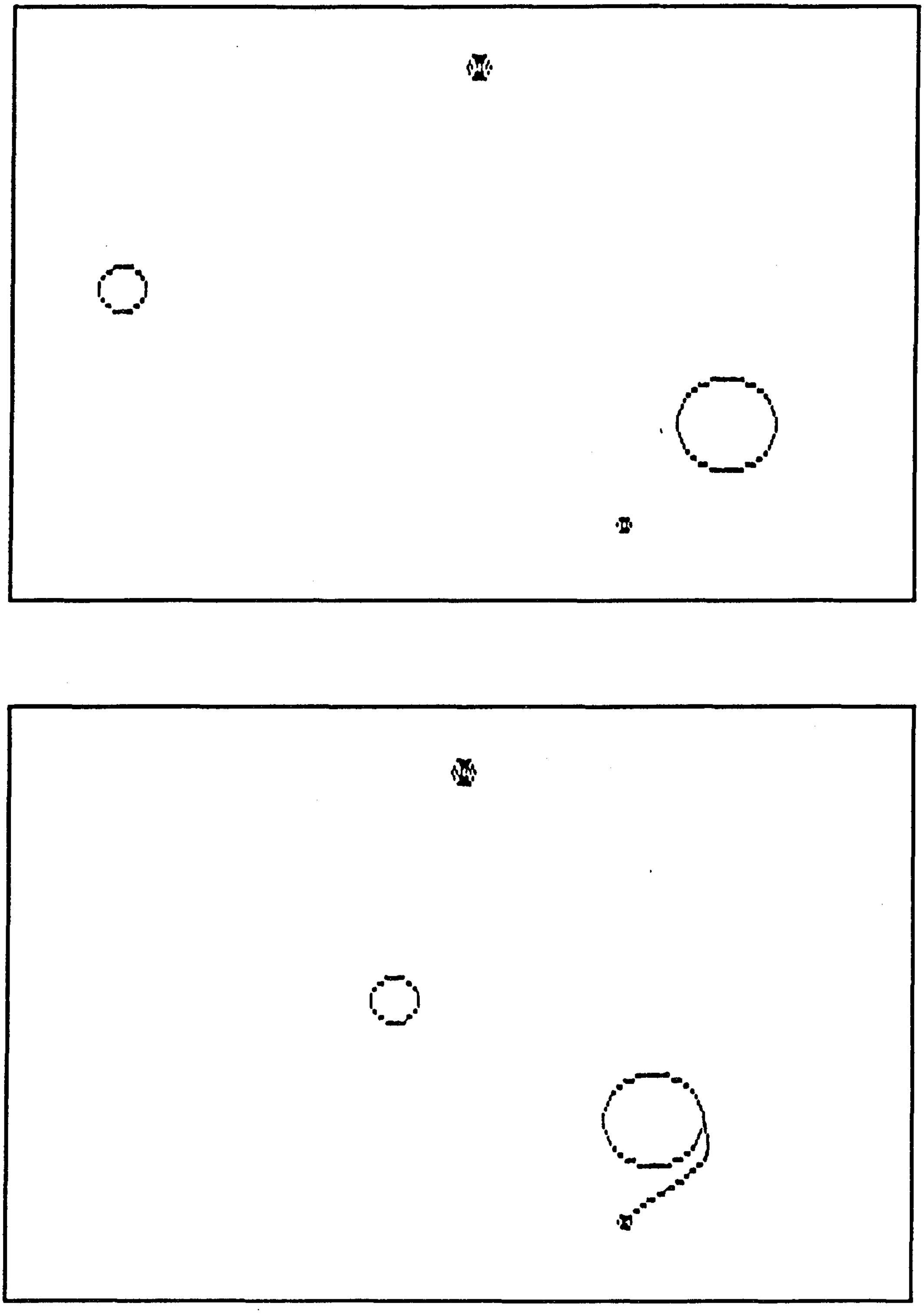


Figure 14. Example two: Robot path with two obstacles.

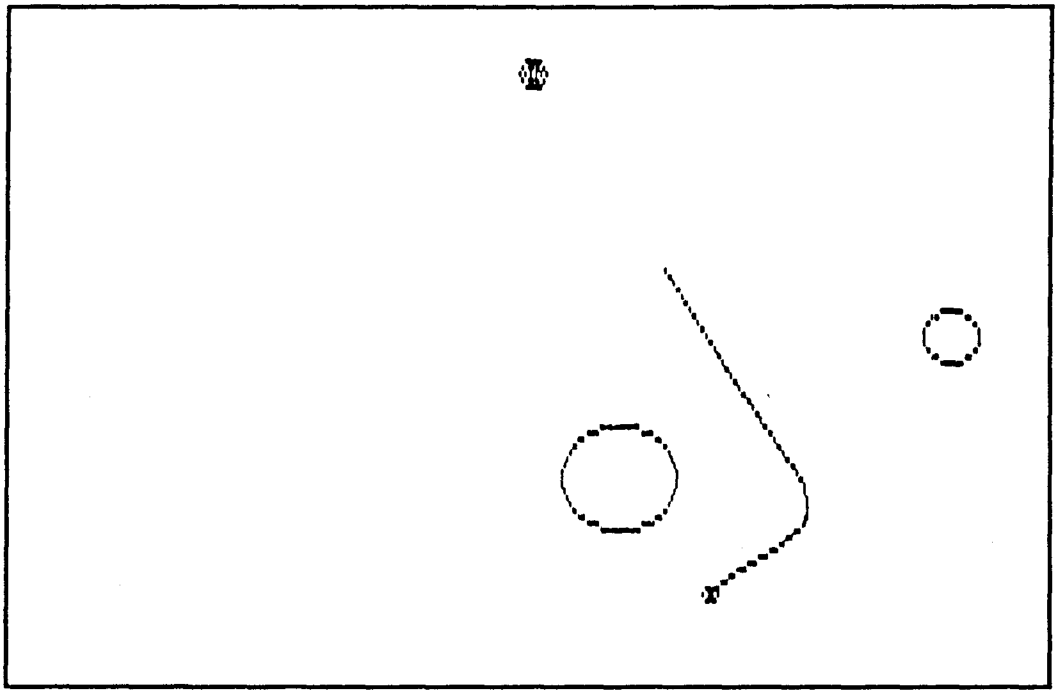


Figure 14, continued.

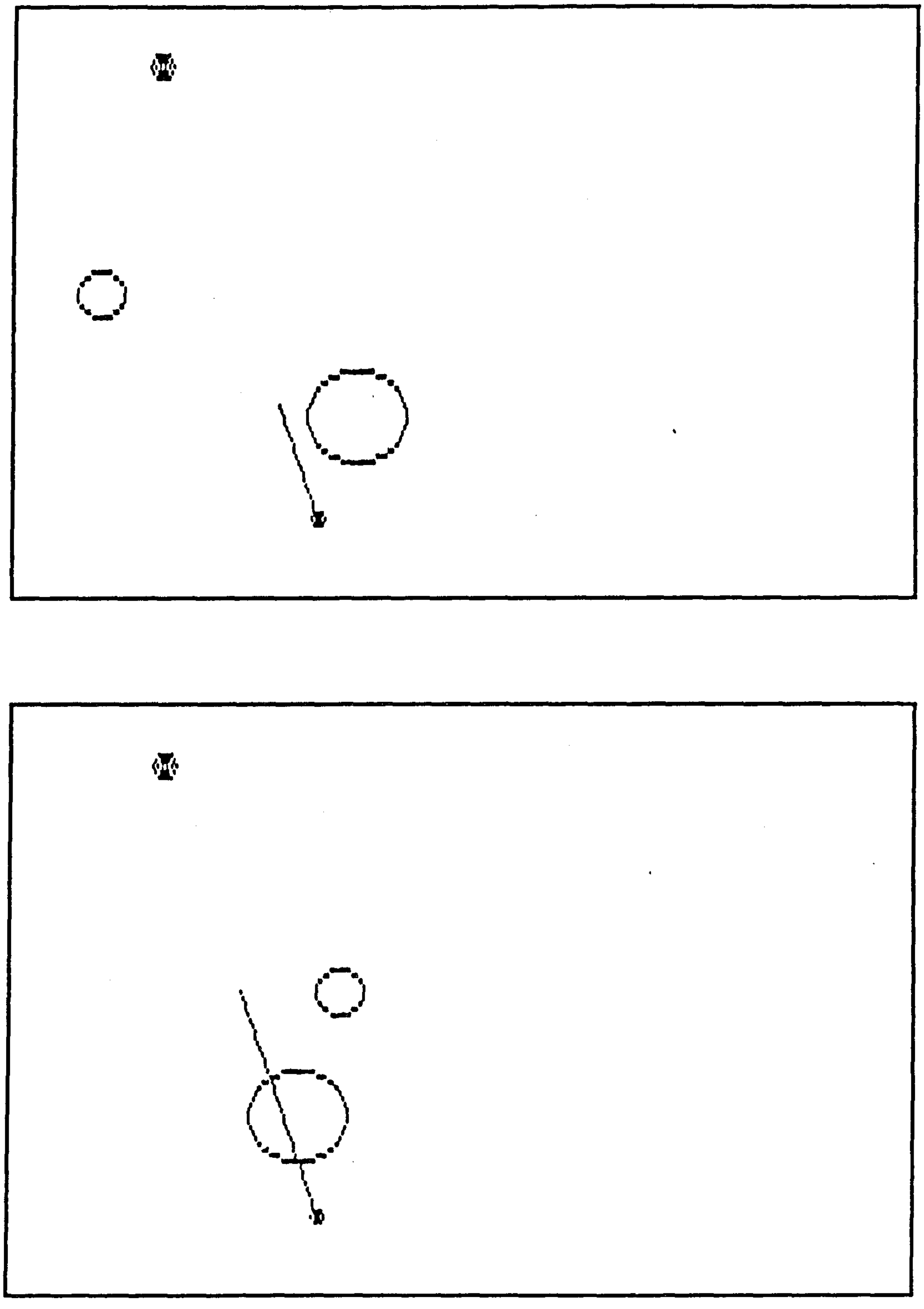


Figure 15. Example three: Robot path directly to goal.

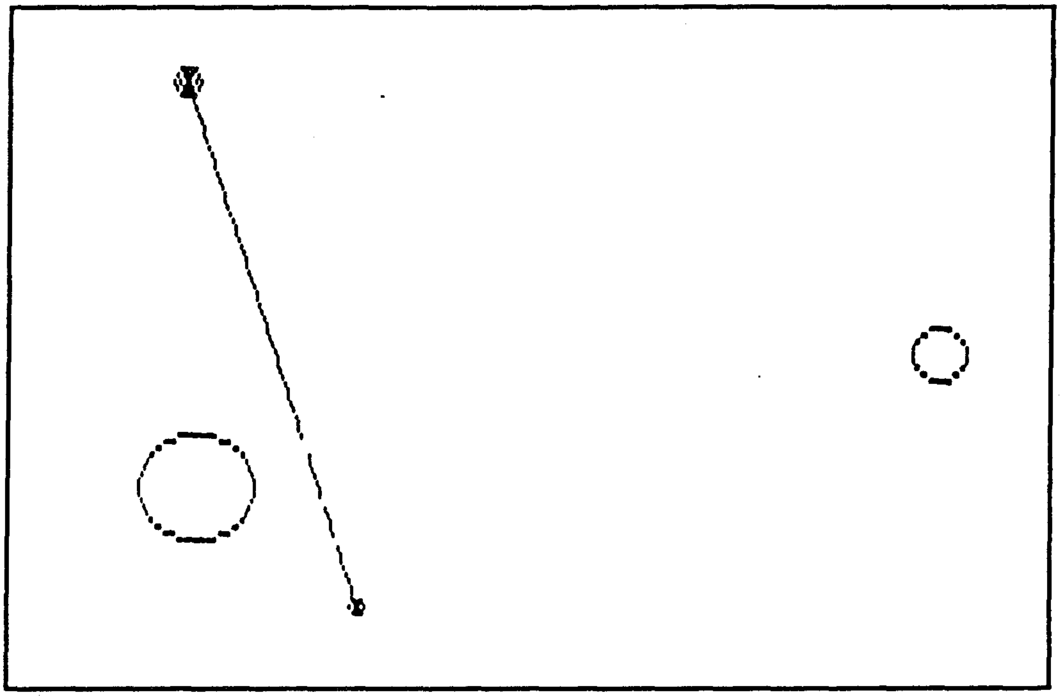
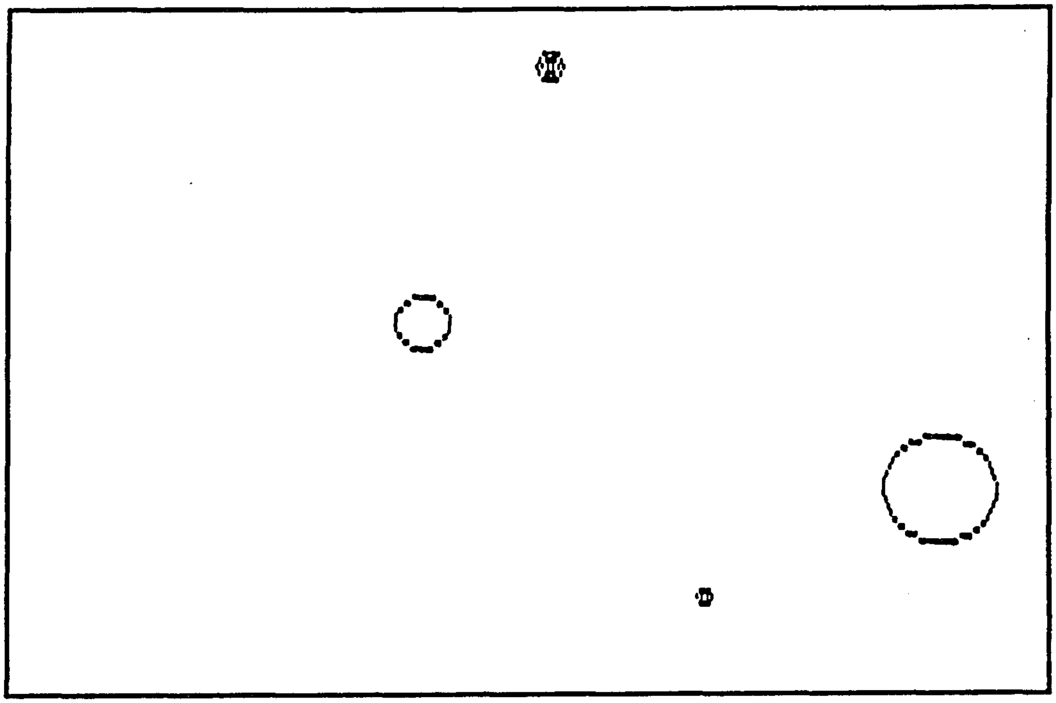


Figure 15, continued.



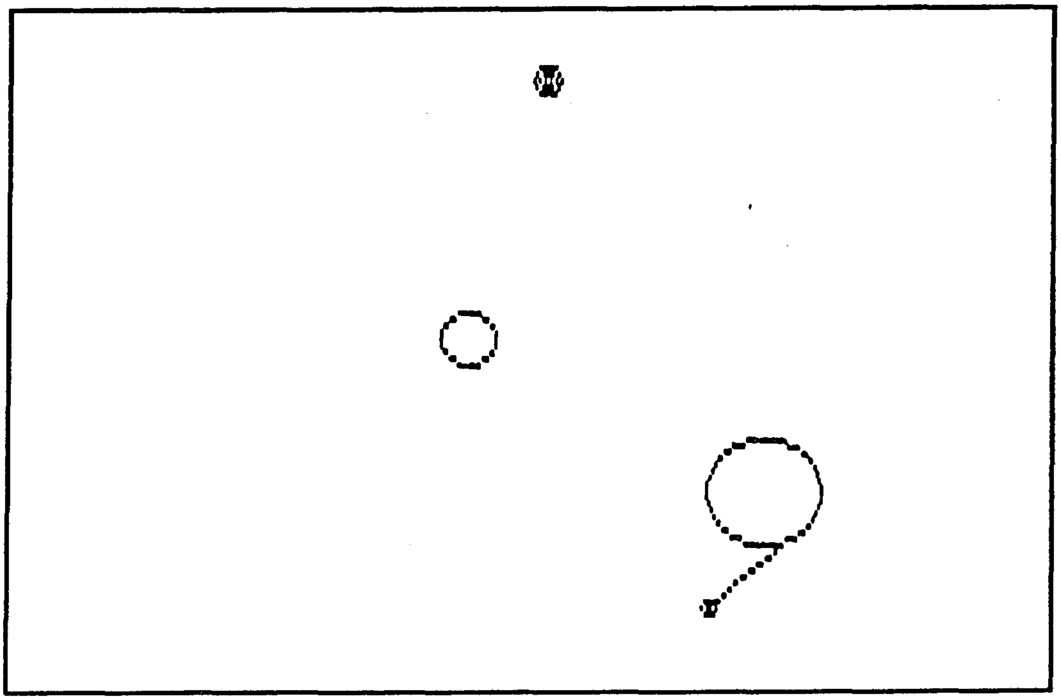
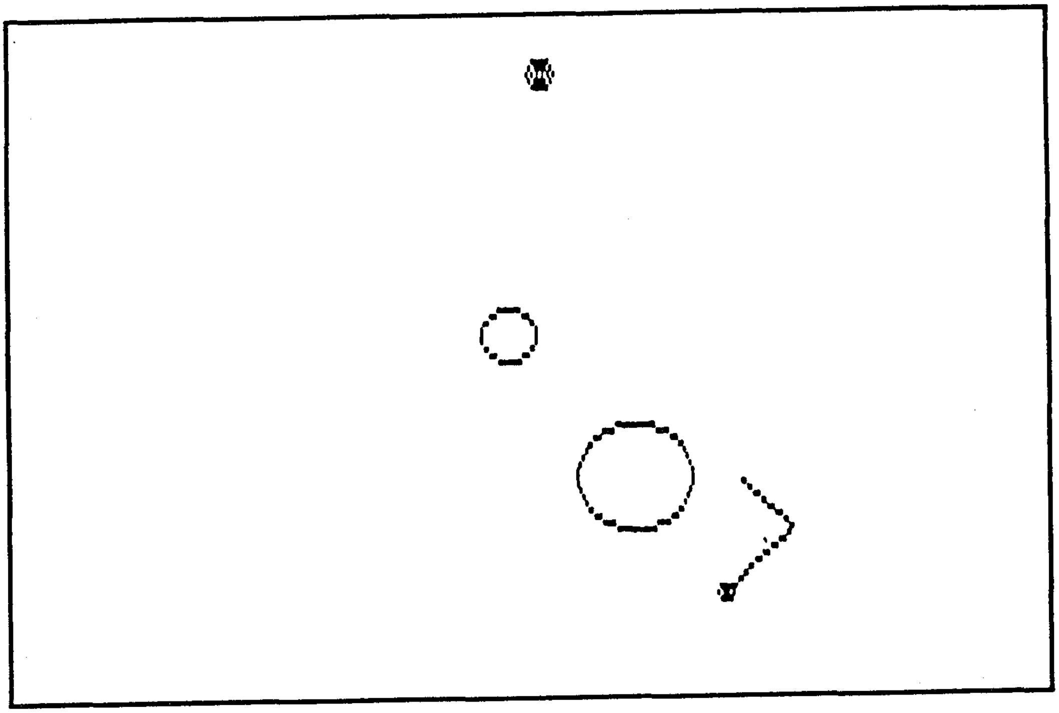


Figure 16. Example four: Robot path with sharp turn.



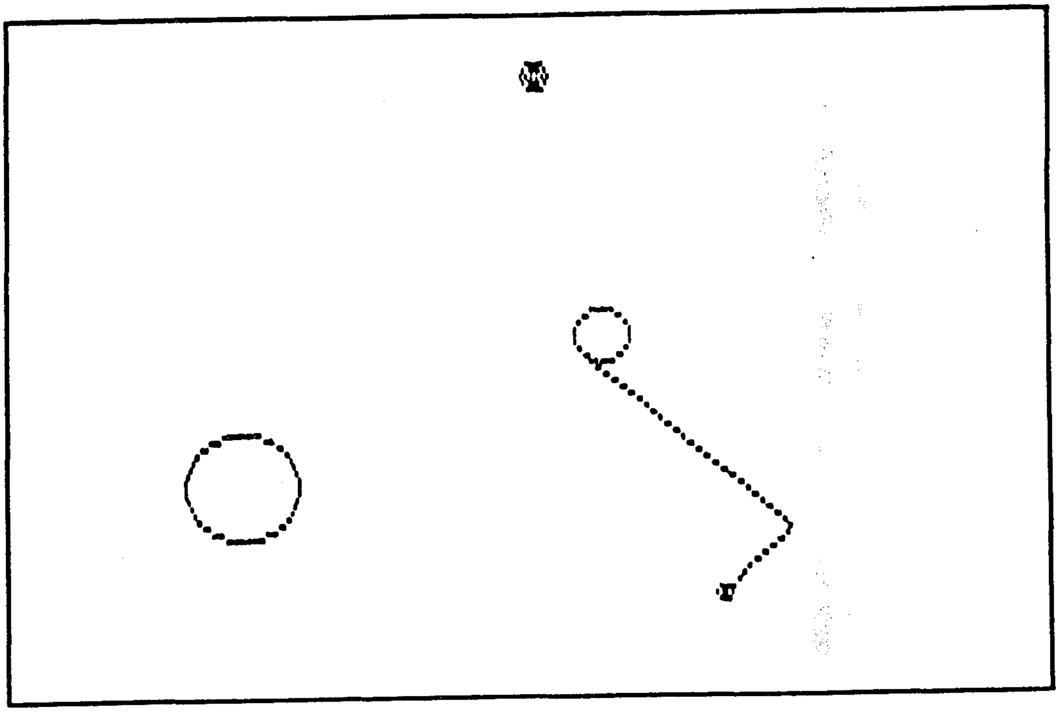


Figure 16, continued.

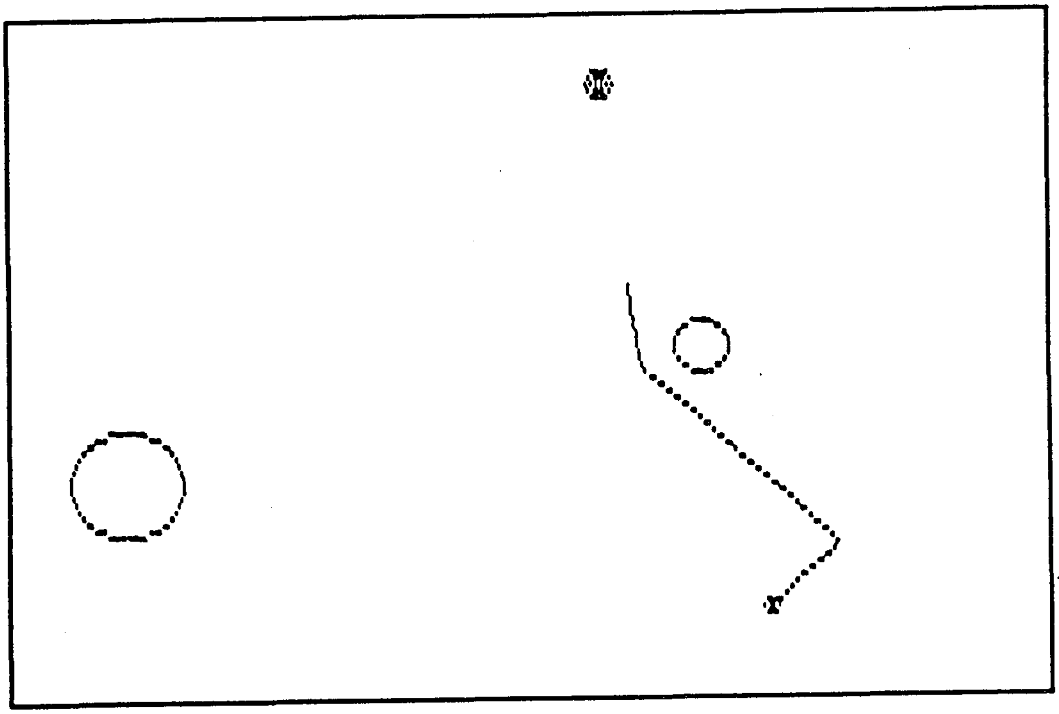
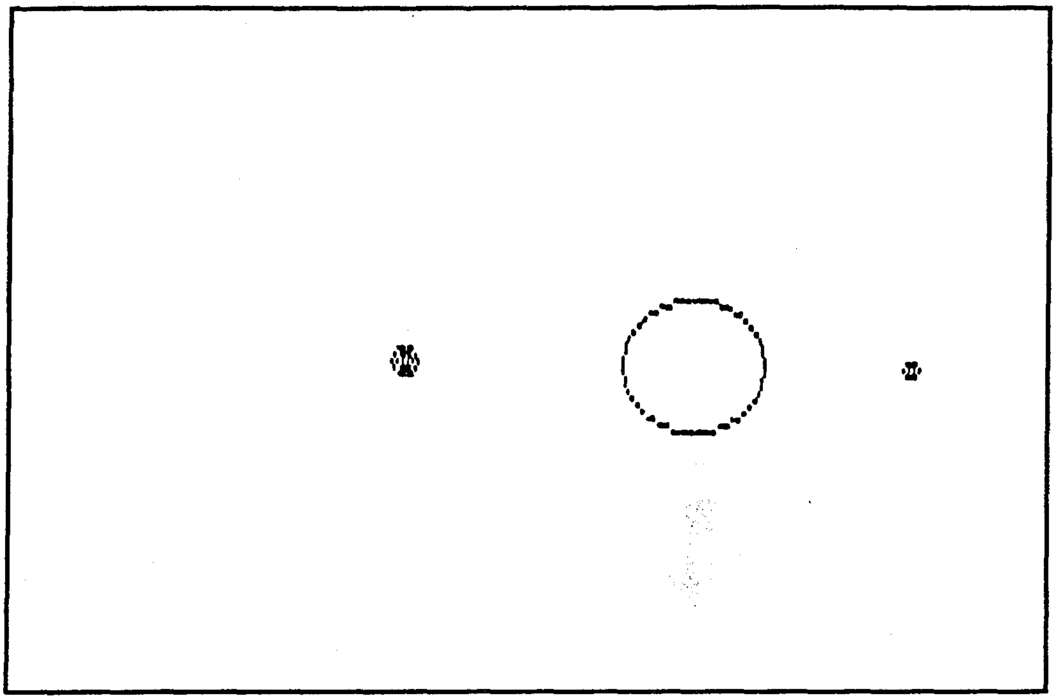


Figure 16, continued.



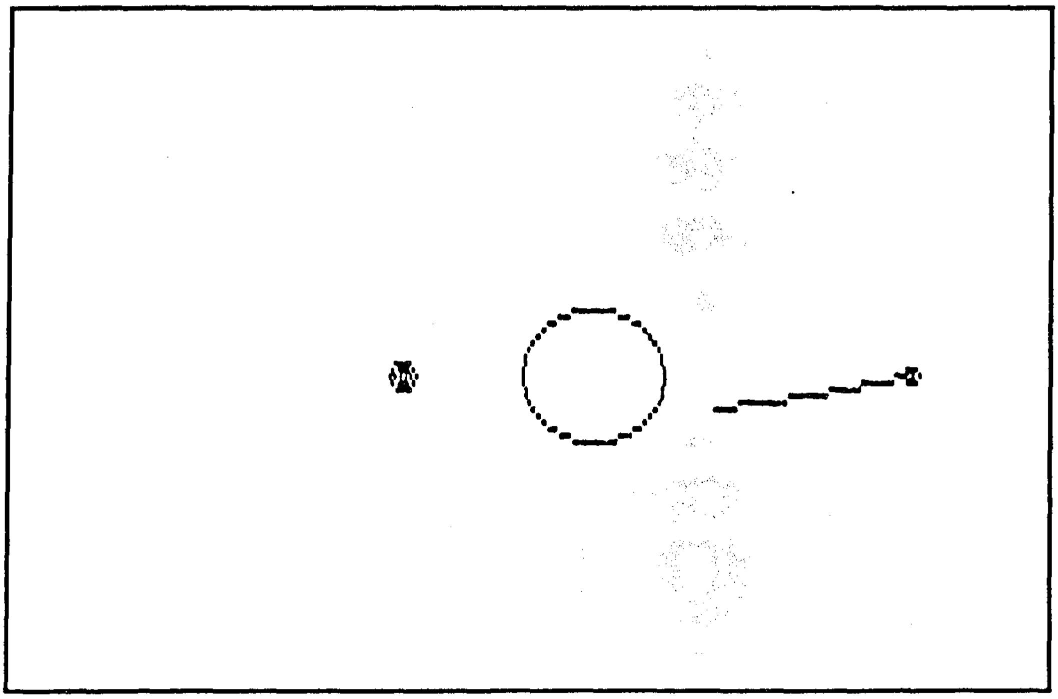
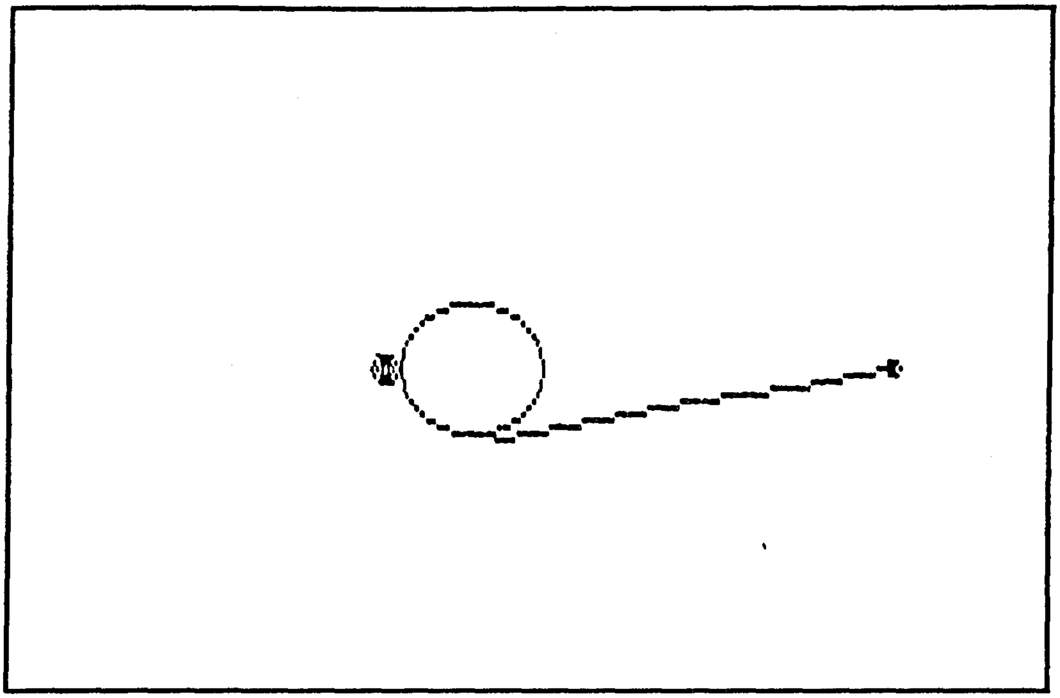


Figure 17. Example five: Robot path curving entirely around obstacle.



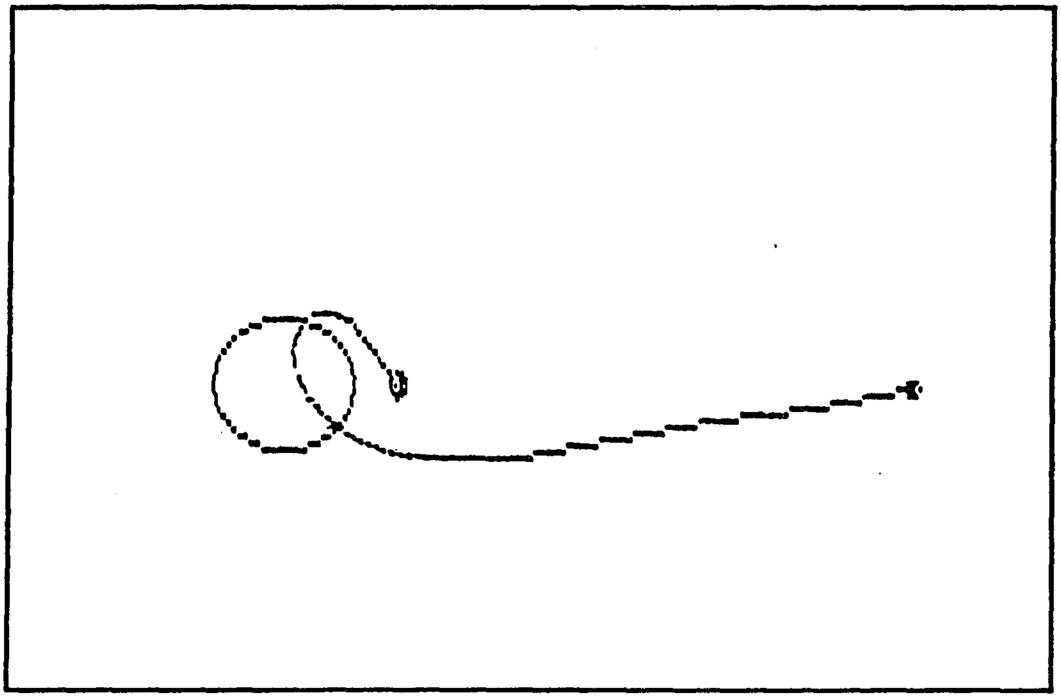


Figure 17, continued.

must actually circle the obstacle until the goal is visible. Although the derived path is optimal for the constant robot speed, a shorter path could be obtained if the robot slowed down to allow the obstacle to clear the goal without collision. Chapter 6 discusses the possibility of extending the algorithm to allow the robot speed to vary.

To illustrate the implementation of the methodology, a complete example of the robot path planning follows. In this example, the robot begins at location $(x_0, y_0) = (3,1)$ with a constant velocity of 2 units per second, and is given the goal (3,15). Four obstacles share the environment with the robot: obstacle I has a configuration space radius of 1 unit, begins at location (4.5,3), and travels at a directional angle of 180 degrees (west) at a speed of 1 unit per second; obstacle II has a configuration space radius of 2 units, begins at location (0,6), and travels at 1 unit per second at an angle of 0 degrees (east); obstacle III has a configuration space radius of 1 unit, begins at location (8,9), and travels at a directional angle of 180 degrees at a speed of 1 unit per second; obstacle IV has a radius of 0.5 units, begins at location (-4,12), and travels at an angle of 0 degrees at 1 unit per second. This starting configuration is illustrated in figure 18.

The first step of the algorithm requires analyzing the visibility of the goal from the robot's current position (3,1). Each of the four obstacles must be examined to determine if they would collide with the robot traveling directly to the goal. Solving equation (12) in section 3.6 for t for obstacle I yields intersection times of 1.5 units and 0.7 units. Since the time required for the robot to reach the goal is 7.0 units, both intersection times lie in the range given in section 3.7:

$$0 < 0.7 < 1.5 < 7.0$$

and thus the goal is hidden.

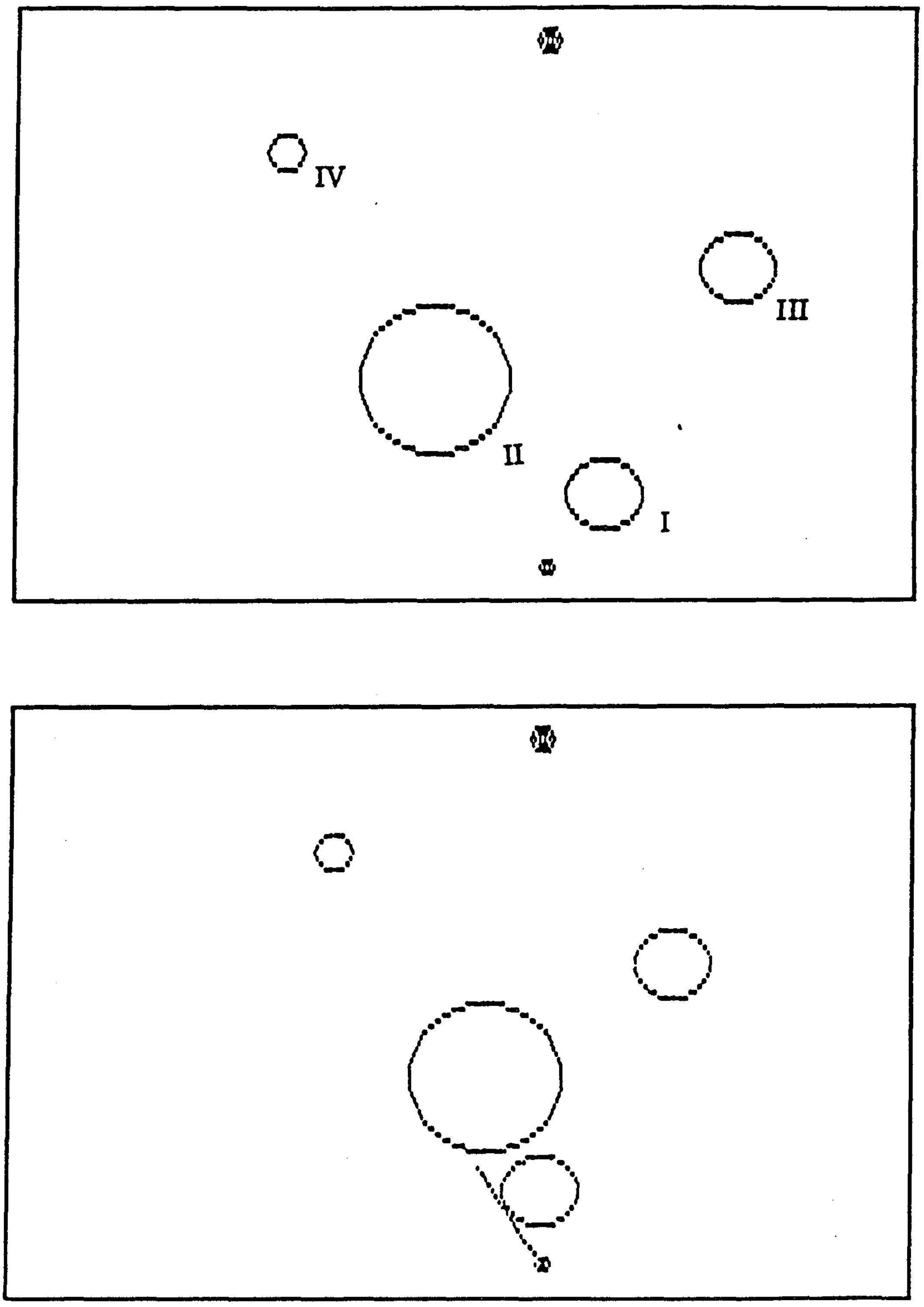
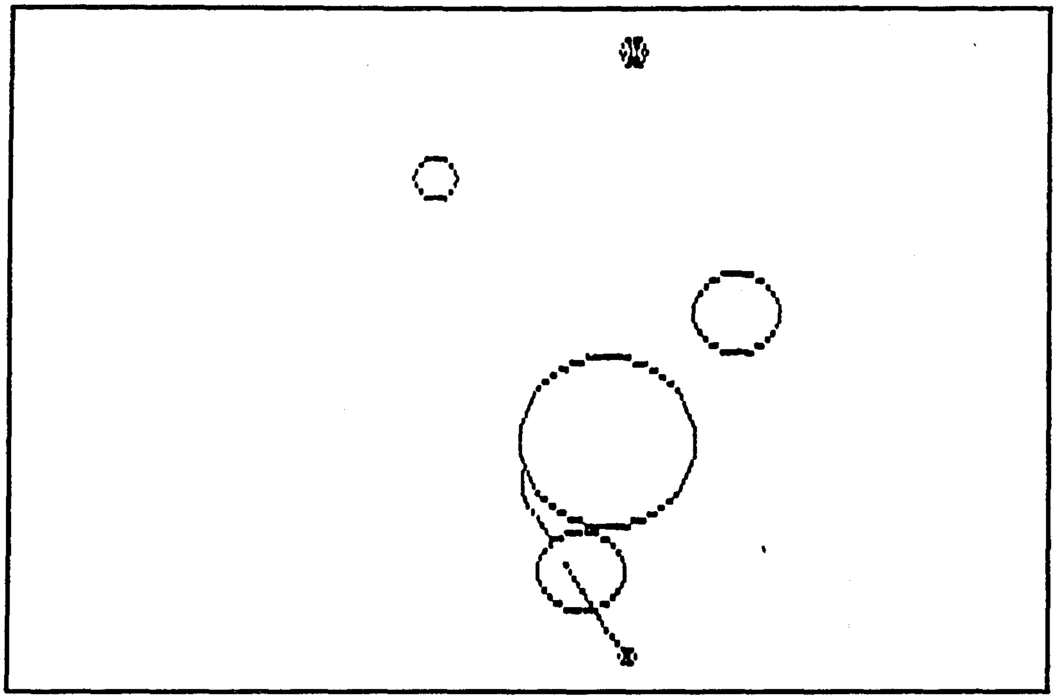


Figure 18. Example six: Robot path with four obstacles.



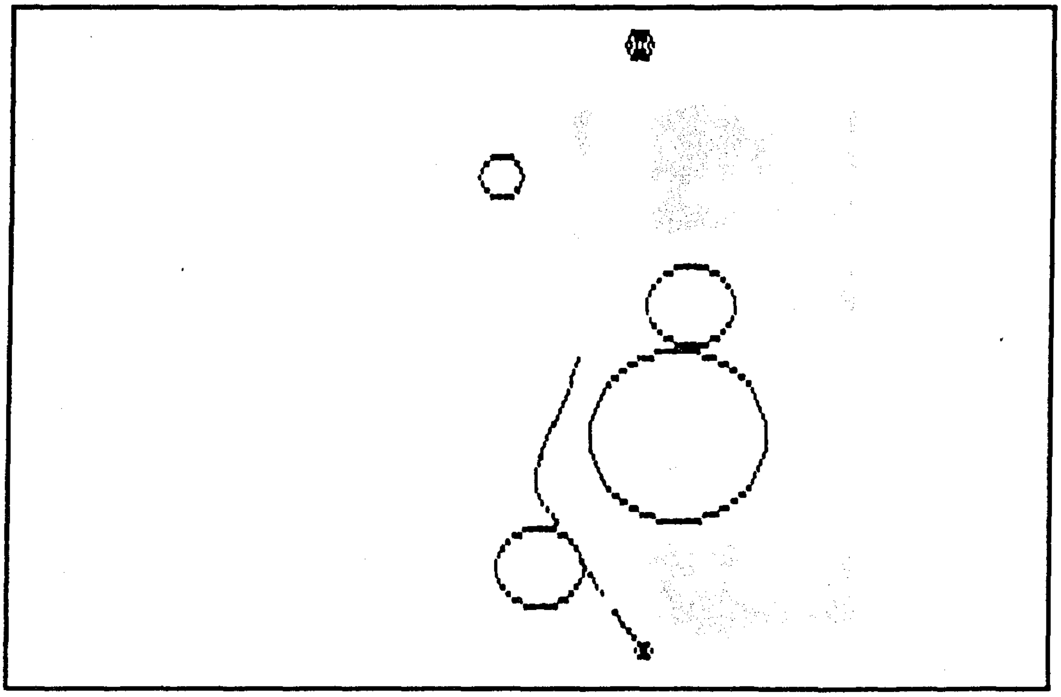
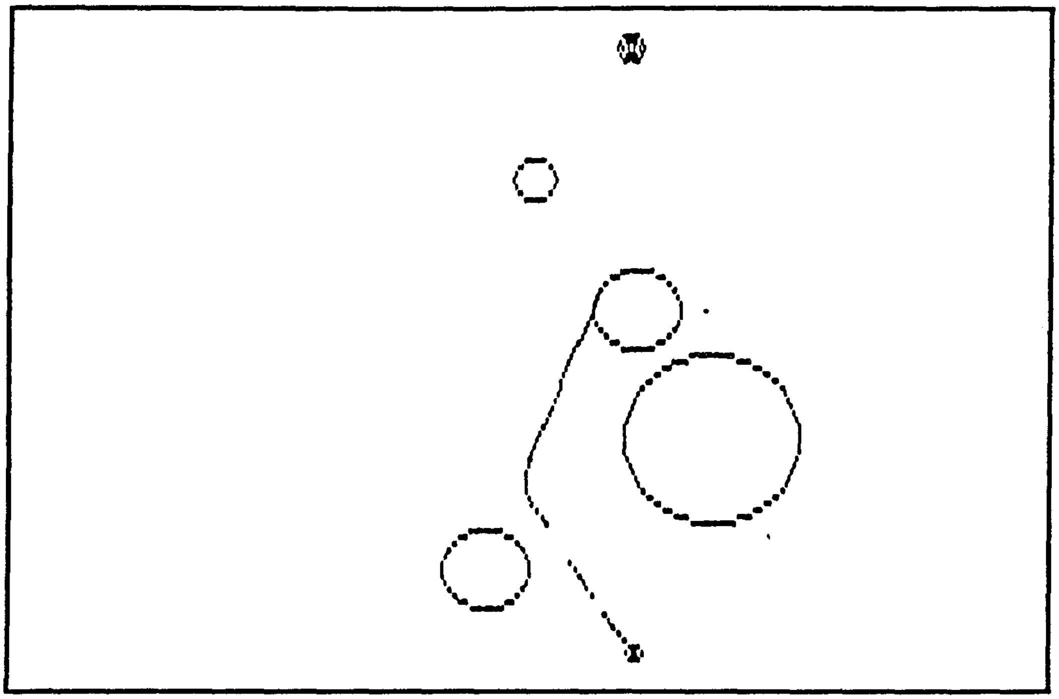


Figure 18, continued.



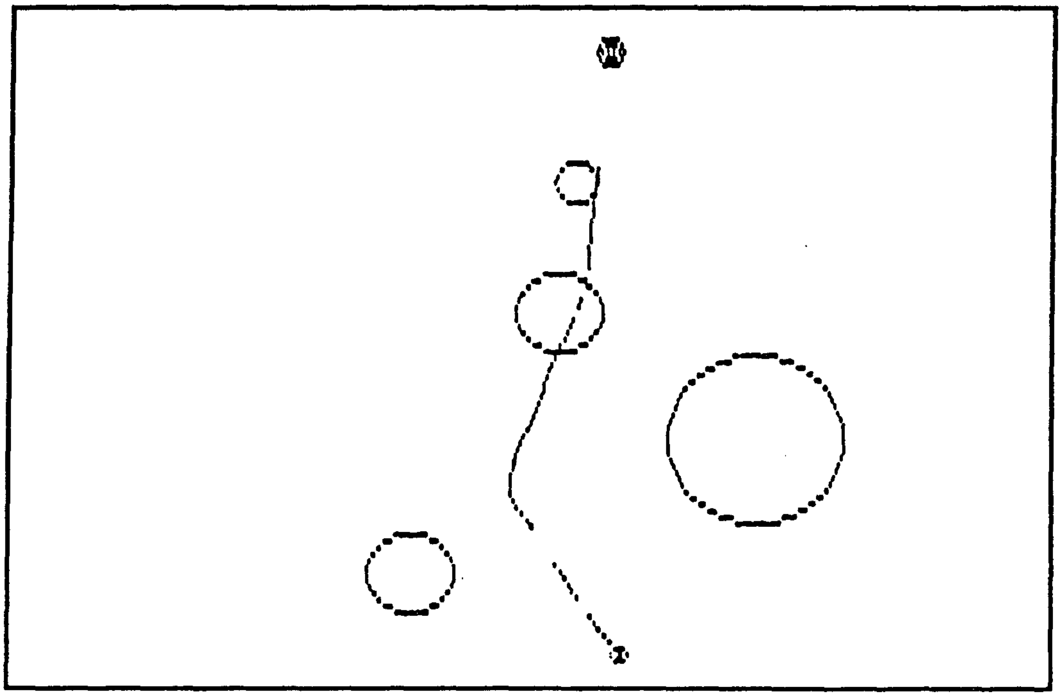


Figure 18, continued.

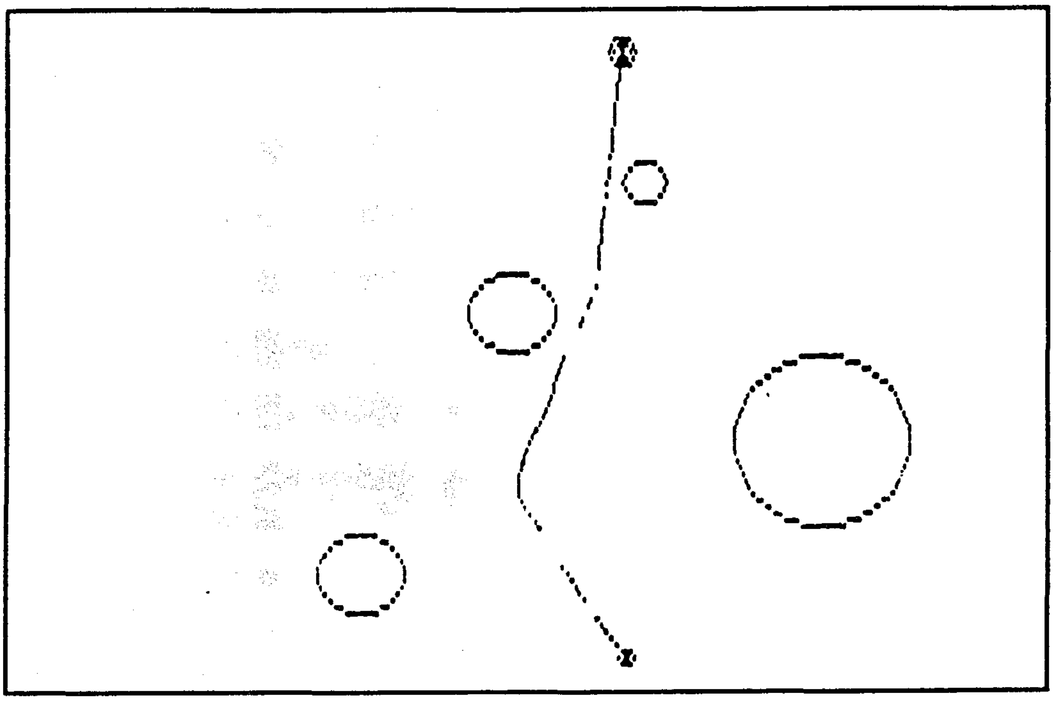


Figure 18, continued.

The second step in the algorithm requires finding the intersection curves between the robot and each obstacle. The current implementation uses a time step δt equal to 0.01, and detects collisions up to a maximum time equal to 1.5 times the time for the robot to reach the goal along a straight-line path from the robot's starting location. The resulting initial intersection curves for this example are shown in figure 19. Calculating the derivative of the intersection curves and comparing the values to the slope of the lines connecting the robot's current position and the intersection points produces the tangent points shown in figure 20. Note that the agreement between the derivative at the selected tangent points and the slope to the robot is quite good. Reducing the size of the time step δt would improve the agreement, but would slow down the search for a path.

The next step is to extend the tangent points which lie on the leading edge of obstacles moving faster than the robot. The tangent which is on the leading edge is tangent (2.367297,3.232057), but since the robot's speed is faster than the speed of this obstacle, an extension of the tangent is not required.

An analysis of the visibility of the tangents reveals that tangent (4.802314,6.386466) cannot be reached without colliding with obstacle I at time 0.602558, which is in the range $[0, 2.839998]$ — the time during which the robot would travel to this tangent. Likewise, tangent (4.966542, 8.382562) cannot be reached without a collision with obstacle I at time 0.611214. The remaining tangents, however, are found to be visible, so new paths are created from those tangents, their costs are computed, and they are added to the set of possible paths in ascending order by cost. Since the time required for the robot to travel to the tangent (4.183071,2.135755) is 0.82 units and the lower bound on the time required for the robot to reach the goal from the tangent is 6.459265, the cost

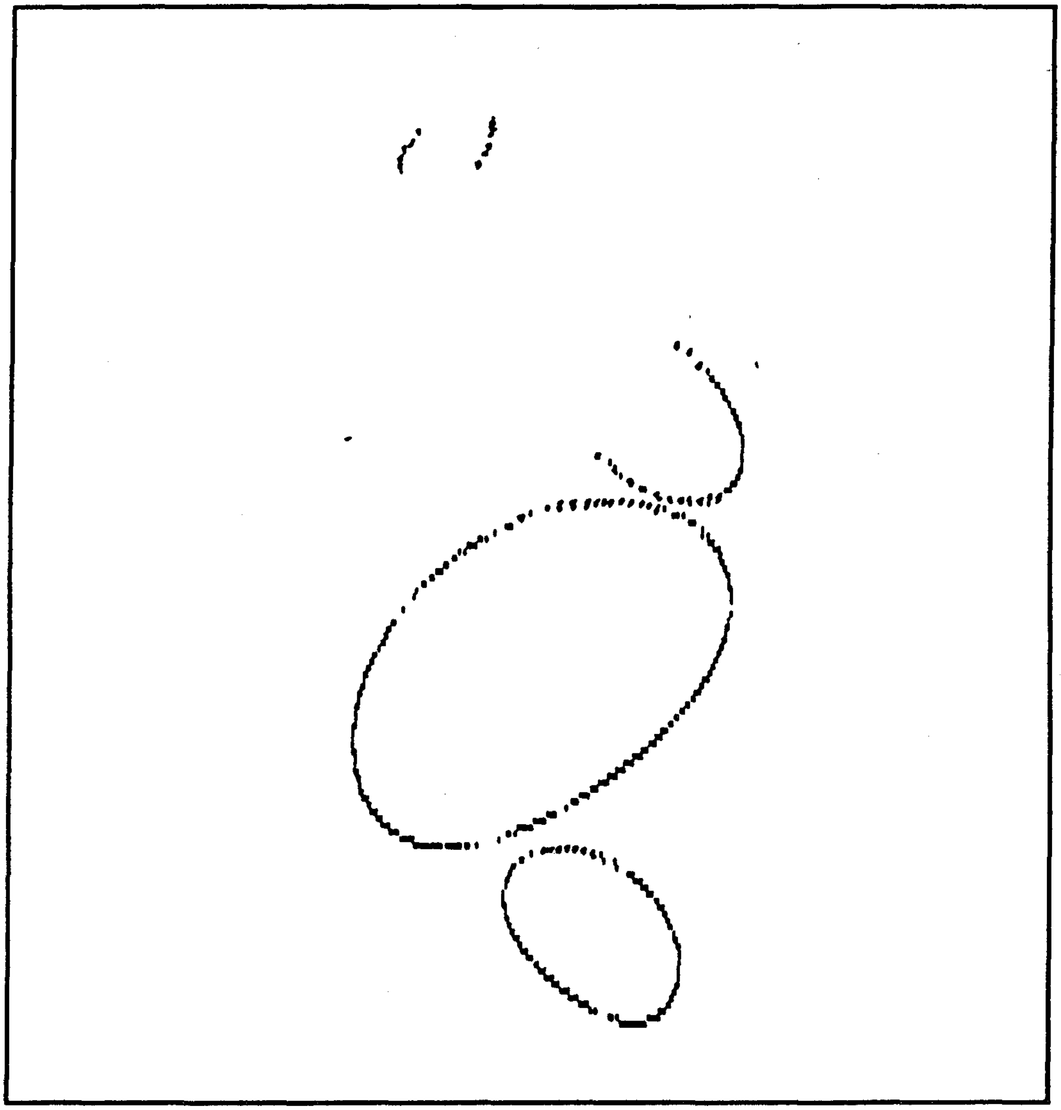


Figure 19. Initial intersection curves for path example six.

Obstacle	Tangent x	Tangent y	Slope difference	Visible?
I	4.183071	2.135755	0.031992	Yes
I	2.367297	3.232057	- 0.033828	Yes
II	0.785481	4.450490	- 0.020413	Yes
II	4.802314	6.386466	0.045894	No
III	4.966542	8.382562	- 0.014439	No

Figure 20. Calculated tangent points for path example six.

assigned to this path is 7.279265. The cost of the remaining paths is computed similarly, resulting in the ordering of paths shown in figure 21.

This process is then repeated, first determining the visibility of the goal from the last vertex of the first path in S, (2.367297, 3.232057), and then finding the visible tangents from this vertex. At the conclusion of the path planning process, the path vertices listed in figure 22 comprise the resulting optimal path. Figure 18 shows a time sequence of the path being executed by the robot. In these figures, the endpoint of the line depicting the robot path is the actual current location of the robot. Note that the robot travels to the front of obstacle I, then curves around the back of obstacle II. The robot next tangentially meets the front edge of obstacle III and turns toward the goal, tangentially passing obstacle IV along the way.

```
cost = 7.052469          elapsed time = 1.159999

      tangent 1 =  (3.000000,  1.000000)
      tangent 2 =  (2.367297,  3.232057)


cost = 7.279265          elapsed time = 0.820000

      tangent 1 =  (3.000000,  1.000000)
      tangent 2 =  (4.183071,  2.135755)


cost = 7.439717          elapsed time = 2.049999
      tangent 1 =  (3.000000,  1.000000)
      tangent 2 =  (0.785481,  4.450490)
```

Figure 21. Order of potential paths for path example six.

x	y	t
3.000000	1.000000	0.000000
0.785481	4.450490	2.049999
0.745201	4.519610	2.089998
0.726966	4.555212	2.109998
0.673500	4.684600	2.179998
0.631406	4.838964	2.259998
0.624206	4.878310	2.279998
0.613112	4.957537	2.319998
0.605208	5.097314	2.389998
0.617665	5.296926	2.489998
0.633515	5.395661	2.539998
0.655308	5.493258	2.589998
0.670912	5.551193	2.619998
0.688707	5.608494	2.649998
0.722244	5.702703	2.699998
0.785694	5.849584	2.779998
2.414878	9.699017	4.869997
3.000000	15.00000	7.536586

Figure 22. Vertices of optimal path for path example six.

CHAPTER 6

CONCLUSIONS/EXTENSIONS

6.1 CONCLUSIONS

In recent years, significant progress has been made towards the development of mobile robots which can perform a variety of tasks autonomously. Many aspects of robot intelligence and control must be addressed to successfully construct a functioning mobile robot. An essential component of the robot intelligence and control must be the navigation algorithm which enables the robot to find its way through its environment without colliding with obstacles. Although numerous approaches to the robot navigation problem have been developed for static environments of non-moving obstacles, substantially less work has been accomplished in the development of algorithms for dynamic environments containing moving objects.

This thesis presents a mathematically-based robot navigation algorithm which has been shown to produce optimal tangent paths for constant robot speeds in a dynamic environment. The extensive literature review performed as part of this thesis revealed no other algorithm which produces optimal paths for a similar scenario. Mathematical representations of the constant-speed obstacles as sheared cylinders and the robot as a cone have been developed and used to derive intersection curves which predict the potential collisions between the robot and the obstacles. The derivative of the intersection curves was formulated and used to locate the points at which the robot and the obstacles meet tangentially. A methodology for selecting visible tangent points was developed, along with an overall planning algorithm using the A* search to plan an optimal path to the goal for a constant robot speed. The implementation of the methodology demonstrated the ability of the algorithm to successfully plan optimal paths in a dynamic environment.

The algorithm developed in this thesis can serve as the central planning module of a robot's navigation system. Actually incorporating this algorithm into a mobile robot would first require the development of an additional module which senses the environment and estimates the current size, speed, and trajectories of the obstacles, providing this data as input to an obstacle growing operation to transform the Cartesian space into configuration space. A second interface would be required to transform the path plans derived by the navigation algorithm into the control signals necessary to actually drive the robot. Additionally, decisions on the monitoring of the path execution must be made to allow replanning following changes sensed in the environment.

6.2 EXTENSIONS

The algorithm described in this thesis produces piecewise-linear paths from tangent point to tangent point until the goal is reached. However, as can be seen in figures 13 through 18, the paths may actually appear to be curved when skirting around obstacles, due to the shortness of these path segments. These path segments are in fact following the intersection curve around the object until a newly selected tangent point pulls the robot path away from that obstacle. While following the intersection curve around an obstacle, a selected tangent point along the obstacle boundary is treated identically to all other tangent points: it is removed from the list of possible paths, expanded to find new tangent points, the costs of the new paths are calculated, and the new paths are added onto the list of potential paths, S. In other words, the A* search procedure is invoked at each step around the obstacle.

However, invoking the A* search at each tangent found while curving around an obstacle may be unnecessary, since the cost of the path will change very little from tangent to tangent. Thus, the likelihood that the path will no longer be the cheapest path while curving around and obstacle is low, resulting in additional overhead and inefficiency when branching back into the A* search. The implementation of this algorithm has verified the existence of this overhead, since curving around obstacles slowed the planning process.

An extension to the existing algorithm would attempt to reduce this overhead by recognizing when a return to the A* search is unnecessary. If the only tangent point produced during an expansion of a node is a tangent which would cause the robot to continue to curve around an obstacle, there is no need to fall back into the A* search. The path should continue to be expanded until the goal is visible or another tangent point off of the current obstacle is detected. At that point, the cost of the path should be calculated, the path should be reinserted into the set of potential paths, S, in ascending order by cost, and the A* search should be reentered. In this manner, excessive overhead can be avoided and the search could potentially be performed quicker. A disadvantage to this extension is that a path which initially appears to be good could be expanded around an obstacle for such a long distance that the path is no longer the path with the least cost. Although the algorithm would eventually fall back into the A* search and recognize that the path was not optimal, processing time would still be wasted. Nevertheless, in the majority of the situations, this extension should save processing time.

Note that this extension cannot apply to situations in which more than one tangent is found during a node expansion. Since the algorithm cannot predict when the robot path should leave the current obstacle, any additional tangents found while curving around the obstacle could belong to the optimal path, and must therefore be saved. Further research is needed to determine if these points of obstacle departure can be predicted, and thus allow some of the intermediate tangent points to be forgotten.

An additional extension to this methodology would allow the constant robot speed restriction to be relaxed, resulting in an algorithm which develops path plans based upon the optimal robot velocities for the current environment. To do this, consider a robot whose speed can vary in a range from v_{min} to v_{max}. The robot representation would then be replaced with a "thick-walled" cone, as illustrated in figure 23. The outer surface of the cone corresponds to the minimum velocity of the robot, v_{min}, while the inner surface of the cone represents v_{max}, the maximum robot velocity.

Allowing the robot speed to vary creates new questions concerning the definition of optimality. For a constant robot speed, a path which is optimal in terms of time will also be optimal in terms of distance. However, paths for a variable-speed robot could be optimal in either time, distance, or even expended energy. Some situations might actually require a combination of several of the constraints to define an optimal path. This increase in the number of degrees of freedom considerably complicates the search for robot paths by causing the size of the search space to multiply rapidly.

For instance, consider the search for a path which navigates the robot to the goal as quickly as possible. This search requires selecting an arithmetic progression of velocities such that $v_i = v_{i-1} + \delta v, i = 0, ..., n$, where $v_0 = v_{min}$ and $v_n = v_{max}$. The paths would be searched using the A* method as described earlier to compute the quickest path to the goal. However, this approach would

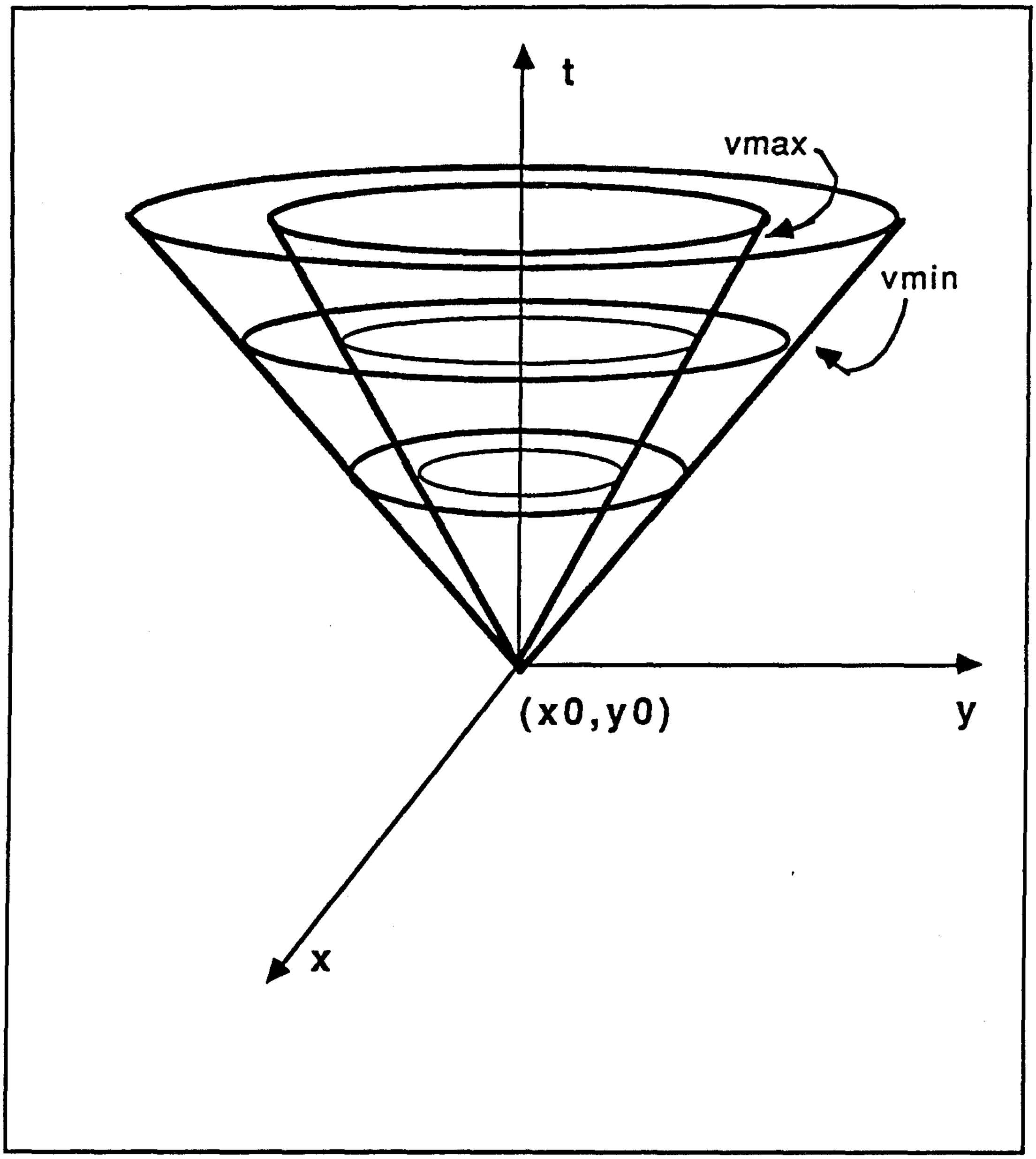
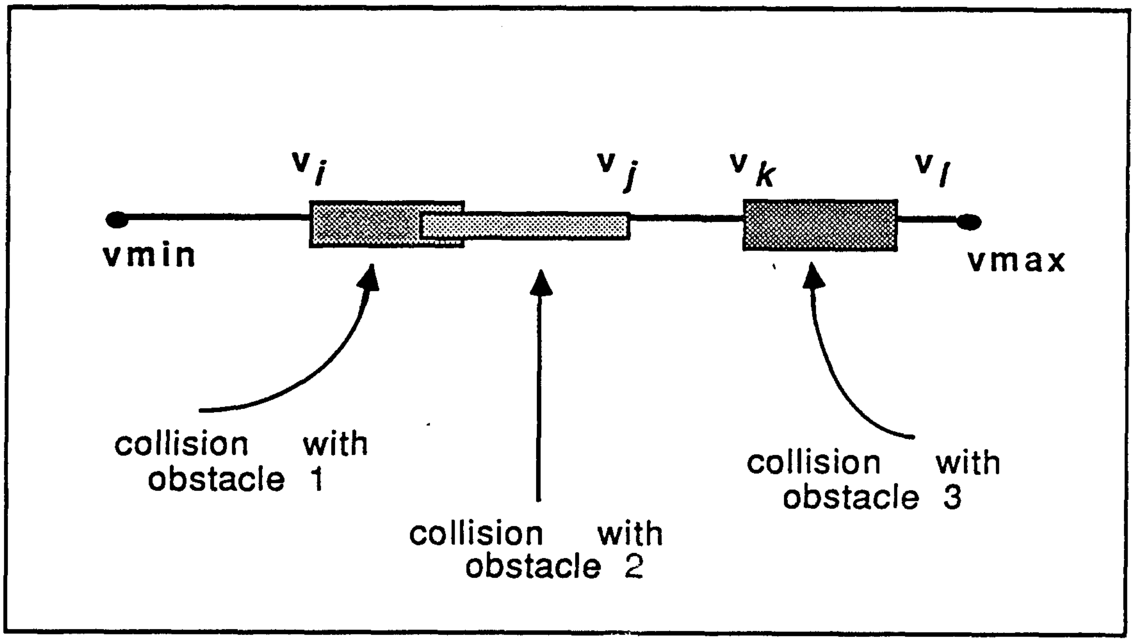


Figure 23. Robot representation for varying speeds.

be severely time-consuming, since for each increment of time, the paths with all increments of velocity from v_{min} to v_{max} must be calculated.

The prohibitive size of the search space may necessitate resorting to heuristic means to derive path plans when the speed of the robot is allowed to vary. Heuristic approaches would use intelligent reasoning about the velocity relationships between the robot and the obstacles to select preferable robot speeds. Such approaches can improve the selected path in either time, distance traveled, or energy expended, but will no longer guarantee an optimal solution. For instance, one such procedure to optimize the path in terms of distance traveled would involve computing the velocities at which the robot would collide with the obstacles if it traveled from its starting location to the goal. Since the linearly moving obstacles will cross the robot's path to the goal at most once, each obstacle will have an associated range of robot velocities at which the two would collide. Figure 24 illustrates a possible scenario for three obstacles in the robot's environment. By selecting a robot velocity in the range from v_{min} to v_i, v_j to v_k, or v_l to v_{max}, the robot can travel directly to the goal without a collision, resulting in the shortest possible path.

Additional heuristics would be required for controlling the selection from a number of valid robot speeds or the selection of a speed when no collision-free path to the goal exists. Likewise, each optimization criteria must have similar heuristics to control the search for a good path based on that optimization constraint. Nevertheless, whatever strategy is selected, it must be evaluated in terms of the nearness of the derived paths to the optimal path. A selected strategy may find paths quickly only at the expense of deriving paths unacceptably far from the optimal path.



v_i
v_j
v_k
v_l
vmin
vmax
collision with obstacle 1
collision with obstacle 2
collision with obstacle 3

Figure 24. Example of ranges of optimal robot speeds.

LIST OF REFERENCES

LIST OF REFERENCES

[1] Abelson, H., and A. diSessa. Turtle Geometry. Cambridge: MIT Press, 1980.

[2] Ahluwalia, R.S., and E.Y. Hsu. "Sensor-Based Obstruction Avoidance Technique for a Mobile Robot." Journal of Robotic Systems 1 (1984): 331-350.

[3] Boyse, John W. "Interference Detection Among Solids and Surfaces." Communications of ACM 22 (January 1979): 3-9.

[4] Brooks, Rodney A. "Solving the Find-Path Problem by Good Representation of Free Space." IEEE Transactions on Systems, Man, and Cybernetics SMC-13 (March/April 1983): 190-197.

[5] Brooks, Rodney A., and Tomas Lozano-Perez. "A Subdivision Algorithm in Configuration Space for Findpath with Rotation." Proceedings of Eighth International Joint Conference on AI. Karlsruhe, West Germany (August 1983): 799-806.

[6] Brooks, Rodney A. "Planning Collision-Free Motions for Pick-and-Place Operations." The International Journal of Robotics Research 2 (Winter 1983): 19-44.

[7] Cahn, David F., and Stephen R. Phillips. "ROBNAV: A Range-Based Robot Navigation and Obstacle Avoidance Algorithm." IEEE Trans. on Systems, Man, and Cybernetics SMC-5 (September 1975): 544-551.

[8] Cameron, Stephen. "A Study of the Clash Detection Problem in Robotics." IEEE International Conference on Robotics and Automation (1985): 488-493.

[9] Charniak, Eugene, and Drew McDermott. Introduction to Artificial Intelligence. Reading: Addison-Wesley Publishing Co., 1985.

[10] Chattergy, R.. "Some Heuristics for the Navigation of a Robot." International Journal of Robotics Research 4 (Spring 1985): 59-66.

[11] Chien, R.T., Ling Zhang, and Bo Zhang. "Planning Collision-Free Paths for Robotic Arm Among Obstacles." IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-5 (May 1983): 91-96.

[12] Cromarty, Andrew S., Daniel G. Shapiro, and Michael R. Fehling. "Still planners run deep: Shallow reasoning for fast replanning." SPIE Applications of Artificial Intelligence 485 (1984): 138-145.

[13] Crowley, James L. "Navigation for an Intelligent Mobile Robot." IEEE Journal of Robotics and Automation RA-1 (March 1985): 31-41.

[14] Denton, Richard V., and Peter L. Froeberg. "Applications of Artificial Intelligence in Automated Route Planning." SPIE Applications of Artificial Intelligence 485 (1984): 126-132.

[15] Doran, James. "Planning and Robots." Machine Intelligence 5. Eds. Bernard Meltzer, and Donald Michie. New York: Edinburgh University Press, 1970.

[16] Fahlman, Scott Elliott. "A Planning System for Robot Construction Tasks." Artificial Intelligence 5 (1974): 1- 49.

[17] Fikes, Richard E., and Nils J. Nilsson. "STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving." Artificial Intelligence 2 (1971): 189-208.

[18] Fikes, Richard E., Peter E. Hart, and Nils J. Nilsson. "Learning and Executing Generalized Robot Plans." Artificial Intelligence 3 (1972): 251-288.

[19] Fikes, Richard E., Peter E. Hart, and Nils J. Nilsson. "Some New Directions in Robot Problem Solving." Machine Intelligence 7. Eds. Bernard Meltzer, and Donald Michie. New York: Edinburgh Univ. Press, 1972.

[20] Gilbert, Elmer G., and Daniel W. Johnson. "Distance Functions and Their Application to Robot Path Planning in the Presence of Obstacles." IEEE Journal of Robotics and Automation RA-1 (March 1985): 21-30.

[21] Gilmore, John F. "The Autonomous Helicopter System." SPIE Applications of Artificial Intelligence 485 (1984): 146-152.

[22] Giralt, Gerges, Ralph Sobek, and Raja Chatila. "A Multi-Level Planning and Navigation System for a Mobile Robot: A First Approach to Hilare." Proceedings of Sixth International Joint Conference on AI, Tokyo (1979): 335-337.

[23] Gladun, V.P. "Decision Planning Systems." Artificial Intelligence and Information-Control Systems of Robots. New York: Elsevier Science Publishers, 1984. 147-151.

[24] Goldstein, Moshe, F.G. Pin, G. deSaussure, and C.R. Weisbin. "3-D World Modeling Based on Combinatorial Geometry for Autonomous Robot Navigation." Proceedings of IEEE International Conference on Robotics and Automation (1987).

[25] Hamel, W.R., S.M. Babcock, and M.G. Hall. "Autonomous Robots for Hazardous and Unstructured Environments." Proceedings of Robots 10 (1986): 5-9 - 5-27.

[26] Hayes-Roth, Frederick, Donald A. Waterman, and Douglas B. Lenat. Building Expert Systems. Reading: Addison-Wesley Publishing Company, 1983.

[27] Howden, W.E. "The Sofa Problem." The Computer Journal 11 (1968): 299-301.

[28] Hunt, Earl B. Artificial Intelligence. New York: Academic Press, Inc., 1975.

[29] Ichikawa, Yoshiaki, and N. Ozaki. "Autonomous Mobile Robot." Journal of Robotic Systems 2 (1985): 135-144.

[30] Iyengar, S.S., C.C. Jorgensen, S.V.N. Rao, and C.R. Weisbin. "Robot Navigation Algorithms Using Learned Spatial Graphs." Robotica 4 (1986): 93-100.

[31] Jentsch, Wiinfried. "Off-line Planning of Collision-Free Trajectories and Object Grasping for Manipulation Robots." Artificial Intelligence and Information-Control Systems of Robots. New York: Elsevier Science Publishers, 1984. 193-196.

[32] Kambhampati, Subbarao, and Larry S. Davis. "Multiresolution Path Planning for Mobile Robots." IEEE Journal of Robotics and Automation RA-2 (September 1986): 135-145.

[33] Khatib, Oussama. "Real-Time Obstacle Avoidance for Manipulators and Mobile Robots." International Journal of Robotics Research 5 (Spring 1986): 90-98.

[34] Kiersey, David M., Joseph S. B. Mitchell, David W. Payton, and Eric P. Preyss. "Multilevel path planning for autonomous vehicles." SPIE Applications of Artificial Intelligence 485 (1984): 133-137.

[35] Koch, E., C. Yeh, G. Hillel, A. Meystel, and C. Isik. "Simulation of Path Planning for a System With Vision and Map Updating." Proceedings of IEEE International Conf. on Robotics and Automation (1985) 146-160.

[36] Koplowitz, Jack, and David Noton. "Motivation System for a Robot." IEEE Trans. on Systems, Man, and Cybernetics SMC-3 (July 1973): 425-428.

[37] Kuan, Darwin T., Rodney A. Brooks, James C. Zamiska, and Mangobinda Das. "Automatic Path Planning for a Mobile Robot Using a Mixed Representation of Free Space." Proceedings of First Conference on AI Applications (1984): 70-74.

[38] Kuan, Darwin T., James C. Zamiska, and Rodney A. Brooks. "Natural Decomposition of Free Space for Path Planning."

Proceedings of IEEE International Conf. on Robotics and Automation (1985): 168-173.

[39] Lamadrid, James. "Avoidance System for Moving Obstacles." SPIE Mobile Robots 727 (1986): 304-311.

[40] Larson, Richard C., and Victor O.K. Li. "Finding Minimum Rectilinear Distance Paths in the Presence of Barriers." Networks 11 (1981): 285-304.

[41] Leighty, Robert D. "Terrain Navigation Concepts for Autonomous Vechicles." SPIE Applications of Artificial Intelligence 485 (1984): 120-125.

[42] Lozano-Perez, Tomas, and Michael A. Wesley. "An Algorithm for Planning Collision-Free Paths Among Polyhedral Obstacles." Communications of the ACM 22 (October 1979): 560-570.

[43] Lozano-Perez, Tomas. "Automatic Planning of Manipulator Transfer Movements." IEEE Transactions on Systems, Man, and Cybernetics SMC-11 (October 1981): 681-698.

[44] Lozano-Perez, Tomas, Matthew T. Mason, and Russell H. Taylor. "Automatic Synthesis of Fine-Motion Strategies for Robots." The International Journal of Robotics Research 3 (Spring 1984): 3-24.

[45] Lumelsky, Vladimir, and Alexander A. Stepanov. "Dynamic Path Planning for a Mobile Automoton with Limited Information on the Environment." IEEE Transactions on Automatic Control AC-31 (November 1986): 1058-1063.

[46] Mandutianu, Dan, and Serban Voinea. "Robots - Skill and Sensitive Behaviour." Artificial Intelligence and Information-Control Systems of Robots. New York: Elsevier Science Publishers, 1984. 229-232.

[47] McDermott, Drew, and Ernest Davis. "Planning Routes through Uncertain Territory." Artificial Intelligence 22 (1984): 107-156.

[48] Meystel, A., and E. Koch. "Computation Simulation of Autonomous Vehicle Navigation." SPIE Applications of Artificial Intelligence 485 (1984): 159-169.

[49] Michie, D. "Formation and Execution of Plans by Machine." Artificial Intelligence and Heuristic Programming. New York: American Elsevier Publishers, 1971.

[50] Mitchell, Joseph S.B. "An Autonomous Vehicle Navigation Algorithm." SPIE Applications of Artificial Intelligence 485 (1984): 153-158.

[51] Moravec, Hans P. "Rover Visual Obstacle Avoidance." Proceedings of International Joint Conference on AI 11 Vancouver (1981): 785-790.

[52] Nilsson, Nils J. Problem-Solving Methods in Artificial Intelligence. New York: McGraw-Hill Book Company, 1971.

[53] Nilsson, Nils J. Principles of Artificial Intelligence. Los Altos: Morgan Kaufmann Publishers, 1980.

[54] O'Dunlaing, C., and C.K. Yap, "The Voronoi Method for Motion-Planning: I. The Case of a Disc." Robotics Research Technical Report No. 53. New York University, Courant Institute of Mathematical Sciences (March 1983).

[55] Parodi, Alexandre M. "A Route Planning System for an Autonomous Vehicle." Proceedings of First Conference on AI Applications (1984): 51-56.

[56] Parodi, Alexandre M. "Multi-Goal Real-Time Global Path Planning for an Autonomous Land Vehicle Using a High-Speed Graph Search Processor." Proceedings of IEEE International Conf. on Robotics and Automation (1985): 161-167.

[57] Petkoff, Boris. "A Domain Independent Framework for Problem Solving." Artificial Intelligence and Information-Control Systems of Robots. New York: Elsevier Science Publishers, 1984. 285- 288.

[58] Pohl, Ira. "Heuristic Ssearch Viewed as Path Finding in a Graph." Artificial Intelligence 1 (1970): 193-204.

[59] Rao, Nageswara, S.S. Iyengar, C.C. Jorgensen, and C.R. Weisbin. "Robot Navigation in an Unexplored Terrain." Journal of Robotic Systems 3 (1986): 389-407.

[60] Rao, Nageswara, S.S. Iyengar, and C.R. Weisbin. "On Autonomous Terrain Model Acquisition by a Mobile Robot." Proceedings of Telerobotics Workshop. California Institute of Technology, January 20-23, 1987.

[61] Rao, Nageswara, S.S. Iyengar, B. John Oommen, and R.L. Kasyap. "Terrain Acquisition by Point Robot Amidst Polyhedral Obstacles." Proceedings of the Third Conference on Artificial Intelligence Applications. (February 1987): 170-175.

[62] Rao, Nageswara, S.S. Iyengar, and G. deSaussure. "The Visit Problem: Visibility Graph-Based Solution." Oak Ridge National Laboratory, CESAR-87/38: 1987.

[63] Raphael, B. "The Frame Problem in Problem-Solving Systems." Artificial Intelligence and Heuristic Programming. New York: American Elsevier Publishers, 1971.

[64] Reif, John H. "Complexity of the Generalized Mover's Problem." Planning, Geometry, and Complexity of Robot Motion. Eds. Jacob

Schwartz, Micha Sharir, and John Hopcroft. Norwood: Ablex Publishing Corporation, 1987.

[65] Sacerdoti, Earl D. "Planning in a Hierarchy of Abstraction Spaces." Artificial Intelligence 5 (1974): 115-135.

[66] Sacerdoti, Earl D. A Structure for Plans and Behavior. New York: Elsevier North-Holland, 1977.

[67] Salter, Richard M. "Planning in a Continuous Domain – An Introduction." Robotica 1 (1983): 85-93.

[68] Sandewall, E.J. "Heuristic Search: Concepts and Methods." Artificial Intelligence and Heuristic Programming. New York: American Elsevier Publishers, 1971.

[69] Schwartz, Jacob T., and Micha Sharir. "On the Piano Movers' Problem: I. The Case of a Two-Dimensional Rigid Polygonal body Moving Amidst Polygonal Barriers." Planning, Geometry, and Complexity of Robot Motion. Eds. Jacob Schwartz, Micha Sharir, and John Hopcroft. Norwood: Ablex Publishing Corporation, 1987. 1-50.

[70] Schwartz, Jacob T., and Micha Sharir. "On the Piano Movers' Problem: II. General Techniques for Computing Topological Properties of Real Algebraic Manifolds." Planning, Geometry, and Complexity of Robot Motion. Eds. Jacob Schwartz, Micha Sharir, and John Hopcroft. Norwood: Ablex Publishing Corporation, 1987. 51-96.

[71] Schwartz, Jacob T., and Micha Sharir. "On the Piano Movers' Problem: III. Coordinating the Motion of Several Independent Bodies: The Special Case of Circular Bodies Moving Amidst Polygonal Barriers." Planning, Geometry, and Complexity of Robot Motion. Eds. Jacob Schwartz, Micha Sharir, and John Hopcroft. Norwood: Ablex Publishing Corporation, 1987. 97-140.

[72] Schwartz, Jacob T., and Micha Sharir. "On the Piano Movers' Problem: V. The Case of a Rod Moving in Three-Dimensional Space Amidst Polyhedral Obstacles." Technical Report No. 83. New York University (October 1983).

[73] Sharir, Micha, and Elka Ariel-Sheffi. "On the Piano Movers' Problem: IV. Various Decomposable Two-Dimensional Motion Planning Problems." Planning, Geometry, and Complexity of Robot Motion. Eds. Jacob Schwartz, Micha Sharir, and John Hopcroft. Norwood: Ablex Publishing Corporation, 1987. 141-153.

[74] Shimura, Masamichi, and Frank H. George. "Rule-Oriented Methods in Problem Solving." Artificial Intelligence 4 (1973): 203-223.

[75] Simon, Herbert A., and Joseph B. Kadane. "Optimal Problem- Solving Search: All-or-None Solutions." Artificial Intelligence 6 (1975): 235-247.

[76] Simon, Herbert A. "Search and Reasoning in Problem Solving." Artificial Intelligence 21 (1983): 7-29.

[77] Stefik, Mark. "Planning with Constraints (MOLGEN: Part 1)." Artificial Intelligence 16 (1981): 111-140.

[78] Stefik, Mark. "Planning and Meta-Planning (MOLGEN: Part 2)." Artificial Intelligence 16 (1981): 141-170.

[79] Stepankova, Olga, and Ivan M. Havel. "A Logical Theory of Robot Problem Solving." Artificial Intelligence 7 (1976): 129- 161.

[80] Tecuci, Gheorghe. "A Framework for Constructing Knowledge-Based Planning Systems." Artificial Intelligence and Information-Control Systems of Robots. New York: Elsevier Science Publishers, 1984. 369-372.

[81] Thorpe, Charles. "Path Relaxation: Path Planning for a Mobile Robot." Proceedings of National Conference on AI (AAAI) (1984): 318-321.

[82] Vere, Steven A. "Planning in Time: Windows and Durations for Activities and Goals." IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-5 (May 1983): 246-267.

[83] Weisbin, Charles, Gerard deSaussure, and David Kammer. "Self-Controlled, A Real-Time Expert System for an Autonomous Mobile Robot." Computers in Mechanical Engineering (September 1986): 12-19.

[84] Whitesides, S.H. "Computational Geometry and Motion Planning." Computational Geometry. Ed. G.T. Toussaint. New York: Elsevier Science Publishers, 1985.

[85] Wilensky, Robert. Planning and Understanding. Reading: Adison-Wesley Publishing Company, 1983.

[86] Wilkins, David E. "Domain-independent Planning: Representation and Plan Generation." Artificial Intelligence 22 (1984): 269-301.

[87] Wilkins, David E. "High Level Planning in a Mobile Robot Domain." Technical Note 388. SRI International (July 15, 1986).

[88] Winston, Patrick Henry. "The MIT Robot." Machine Intelligence 7. Eds. Bernard Meltzer, and Donald Michie. New York: Edinburgh University Press, 1972.

[89] Winston, Patrick Henry. Artificial Intelligence. Second Edition. Reading: Addison-Wesley Publishing Co., 1984.

[90] Wong, E.K., and K.S. Fu. "A Hierarchical-Orthogonal-Space Approach to Collision-Free Path Planning." Proceedings of IEEE International Conf. on Robotics and Automation (1985): 506-511.

[91] Zlatareva, Nely P. "A Plan Generating System of an Intellectual Robot Using a Frame Representation of the Information." Artificial Intelligence and Information- Control Systems of Robots. New York: Elsevier Science Publishers, 1984. 399-403.

VITA

Lynne E. Parker was born in Knoxville, Tennessee on April 5, 1961. She attended elementary and junior high schools in that city and was graduated as valedictorian from Powell High School in June 1979. The following September she entered Tennessee Technological University, and in June 1983 she graduated summa cum laude with a Bachelor of Science degree in Computer Science. Immediately upon graduation, she began work at Martin Marietta Energy Systems (formerly known as Union Carbide Corporation) in Oak Ridge, Tennessee.

While continuing her employment at Martin Marietta, she entered The University of Tennessee as a part-time student to study toward a Master's degree in Computer Science. In the fall of 1986, she joined the research staff at the Oak Ridge National Laboratory. Upon graduation in June 1988, she will continue employment at the Oak Ridge National Laboratory.

The author is a member of Phi Kappa Phi, Omicron Delta Kappa, Kappa Mu Epsilon, and the Association for Computing Machinery.