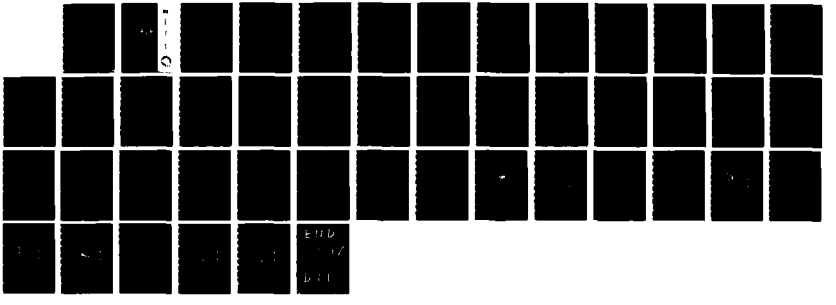
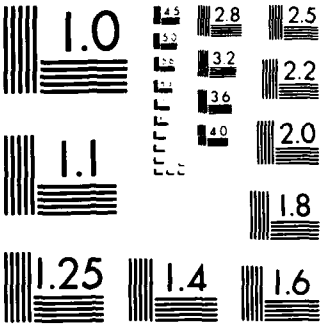
AD-A184 875 TWO DIMENSIONAL PATH PLANNING WITH OBSTACLES AND SHADOWS(U) MARYLAND UNIV COLLEGE PARK CENTER FOR AUTOMATION RESEARCH S PURI ET AL JAN 87 CAR-TR-255 1/1

UNCLASSIFIED ETL-0452 DACA76-84-C-0004 F/G 12/9 NL





ETL-0452

Two dimensional path planning with obstacles and shadows

Sunil Puri
Larry S. Davis

Center for Automation Research
University of Maryland
College Park, MD 20742

January 1987

Prepared for
U.S. ARMY CORPS OF ENGINEERS
ENGINEER TOPOGRAPHIC LABORATORIES
FORT BELVOIR, VIRGINIA 22060-5546

E T L

Destroy this report when no longer needed.
Do not return it to the originator.

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

The citation in this report of trade names of commercially available products does not constitute official endorsement or approval of the use of such products.

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

AD-A184075

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS N/A
2a. SECURITY CLASSIFICATION AUTHORITY N/A	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) CAR-TR-255 CS-TR-1760	5. MONITORING ORGANIZATION REPORT NUMBER(S) ETL-0452
6a. NAME OF PERFORMING ORGANIZATION University of Maryland 6b. OFFICE SYMBOL (If applicable) N/A	7a. NAME OF MONITORING ORGANIZATION Army Engineer Topographic Laboratories
6c. ADDRESS (City, State and ZIP Code) Center for Automation Research College Park, MD 20742	7b. ADDRESS (City, State and ZIP Code) Fort Belvoir, VA 22060
8a. NAME OF FUNDING/SPONSORING ORGANIZATION N/A 8b. OFFICE SYMBOL (If applicable) N/A	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER DACA76-84-C-0004
8c. ADDRESS (City, State and ZIP Code) N/A	10. SOURCE OF FUNDING NOS. PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT NO.
11. TITLE (Include Security Classification) Two Dimensional Path Planning with Obstacles and Shadows	
12. PERSONAL AUTHOR(S) Sunil Puri and Larry S. Davis	
13a. TYPE OF REPORT Technical 13b. TIME COVERED FROM _____ TO N/A	14. DATE OF REPORT (Yr., Mo., Day) January 1987 15. PAGE COUNT 44
16. SUPPLEMENTARY NOTATION	
17. COSATI CODES FIELD / GROUP / SUB. GR.	18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

A mobile robot navigates with a limited knowledge of its environment because of the restricted field of view and range of its sensors, and the occlusion of parts of the environment in any single image. Most path planning algorithms consider only free regions and obstacles in the robot's world for path planning. The objective of this report is to extend the classical path planning paradigm to include occluded regions. This introduces the new problem of deciding when (or whether) to employ the sensor system during the execution of the path to, potentially, reveal the occluded regions as obstacles or free space for the purpose of replanning.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS ☐	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE NUMBER (Include Area Code) 22c. OFFICE SYMBOL

DD FORM 1473, 83 APR EDITION OF 1 JAN 73 IS OBSOLETE

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

1. INTRODUCTION

Given a robot's initial and final goal locations, the problem of mobile robot navigation is to find an appropriate path from an initial position to a goal position such that the mobile robot can smoothly travel through the area avoiding obstacles.

A mobile robot navigates with a limited knowledge of its environment because of the restricted field of view and range of its sensors, and because of the occlusion of parts of the environment in any single image. As a result the problem of reaching a final goal point has to be treated as a sequence of local problems. This will involve generation of subgoals and planning local paths to this sequence of subgoals which have to be attained successively in order to reach the final goal. In this report, our objective is to find a locally optimal path from the current position of the robot to the selected subgoal. The problem of generation of subgoals will not be considered.

A mobile robot's path planning space can be decomposed into free regions, obstacles, occluded regions and unexplored regions. Most path planning algorithms consider only free regions and obstacles in the robot's world for path planning. The objective of this report is to extend the classical path planning paradigm to include occluded regions; this introduces the new problem of deciding when (or whether) to employ the vision system during the execution of the path to, potentially, reveal the occluded regions as obstacles or free space for the purpose of replanning.

For the purpose of solving this problem the robot's path planning space is modeled by a plane containing polygonal obstacles. A height specification is associated with each polygon in order to outline the occluded regions in the robot's world. Simulation of the robot's vision system is done under constraints of field of view and range of vision for the purpose of classifying the robot's environment into free, obstacle, occluded and unexplored regions. Section 2 of the report describes the world model and the algorithm used for vision simulation.

Section 3 describes a path planning process which uses a multiresolution representation path planning algorithm described in Kambhampati and Davis [Kambh86] to plan a negotiable path. A cost function is developed which estimates how close this negotiable path is to optimal. A decision procedure is employed to decide how far to travel along a path before employing the vision sensor module again to potentially map the occluded regions in the shadows of obstacles in the hope of discovering a lower cost path by replanning.

Section 4 gives results of the implementation of vision simulation and the path planner.

1.1. Multiresolution Representation Based Path Planning

[Kambh86] describes a number of algorithms for mobile path planning based on a multiresolution representation of the robot's immediate environment. The multiresolution representation used is the quadtree [Samet83]. A quadtree is a recursive decomposition of a binary array into uniformly colored 2^i x 2^i blocks. Free regions are represented by white nodes and obstacle regions by black nodes.

Since a mobile robot will traverse any given path only once, our path planner tries to develop a negotiable path quickly rather than develop an optimal path, which is a costly operation.

The path planning algorithm is a simple A^* search algorithm with the evaluation function, f, defined as follows :

$$f = g + h$$

Here g is the distance of the current node in the search from the start node, and h is a heuristic estimate of the goodness of the remainder of the path P. The heuristic h is the difference of two components, h_g and h_d, where h_d is the distance of the nearest obstacle from the current node, which determines by how far the resultant path will avoid obstacles, and h_g is the straight line distance between the current node and the goal. The result of applying the A^* algorithm to the quadtree is a list of nodes from the quadtree which define a path between the start node and the goal.

Staged search is an important extension of this simple path planning algorithm involving the ability to deal with grey nodes in the quadtree (nonterminal nodes which have both black and white descendants). Dealing with grey nodes can greatly reduce the number of blocks that need to be considered in building an initial estimate of a path, resulting in computational savings.

2. WORLD MODELING

In order to perform the task of avoiding obstacles and trying to reach a goal location, the robot must build and maintain a model of its environment in which features that affect its movement are represented. The robot will be equipped with a sensor and will attempt to model the environment from its perceptions. Though the world in which the robot must operate is three-dimensional, it moves in a two-dimensional ground plane only. The result of the analysis of its perceptions will be used to create a model in which regions on the ground plane are classified by traversability. This is adequate for path planning and allows a two dimensional representation of all types of obstruction. Also in order to maintain the world model, information available from a new viewpoint must be combined with that already in the model to produce a new world description.

The section below describes simulation of perception by a vision sensor to classify the world into free space, obstacles, occluded and unexplored regions. The vision sensor is of the ranging type so that the relative locations and distances of obstacles from the robot are made available in the simulation. Assuming that the obstacles in the robot's environment are convex polyhedra which are in contact with the ground plane, the artificial world to be used for path planning can be constructed by projection of these polyhedral obstacles into polygons in the two-dimensional ground plane. A height-of-feature is associated with each polygon in order to determine the occluded regions of the obstacles. The simulation also assumes that the robot knows its position in the environment exactly. Information available from each viewpoint after the simulation is also integrated

into a map of the environment.

2.1. Algorithm for Vision Simulation

This section describes an algorithm for determining visible line segments and occluded regions of a set of polygons, each associated with a height parameter, which represent obstacles in our world. The vision simulation is done under constraints based on the range of vision and field of view of a sensor located at a point and pointing in a particular direction.

2.1.1. Description of the World

The world under consideration lies in the Cartesian plane and consists of a set of polygons $\{P_i\}$, i = 1, 2, ... , n in the plane. Each polygon P_i is defined by a list of vertices $V_{i_j} = (x_{i_j}, y_{i_j})$, $j = 1, 2, ..., m$. We define the vertex V_{i_j} to be the jth vertex of the ith polygon. It is also assumed that the polygons are convex and disjoint. The vertices of the polygons are given in clockwise order.

A polygon P_i with vertices V_{i_j} can also be represented by a sequence of line segments $V_{i_1}V_{i_2}$, $V_{i_2}V_{i_3}$, ... $V_{i_m}V_{i_1}$, which we define as the set of polygon edges $\{L_{i_j}\}$, j = 1, 2, ..., m. L_{i_j} is the jth edge of the ith polygon defined by the vertices $V_{i_j}V_{i_{j+1}}$.

In addition, each polygon P_i has a height specification h_i associated with it, representing a measure of height of the obstacle.

2.1.2. Definitions of Terms

We define a *view* v to be an ordered pair consisting of a point p and an angle ϕ. The point p defines the location and the angle ϕ defines the viewing direction of the vision sensor in the world. The viewing direction ϕ is specified with respect to the x axis of the Cartesian plane defining the world with point p as the origin. The point p cannot lie inside a polygon.

A *line of sight* is defined to be the ray originating at $p_0 = (x_0, y_0)$, the location of the sensor, and passing through some point $q = (x, y)$. A line of sight may also be defined as the ray which originates at p_0 and makes some angle ψ with respect to the x-axis of the Cartesian plane with the location of the sensor at the origin,

$$\psi = \arctan ((y - y_0) / (x - x_0))$$

The *angular visibility limit*, AL, is defined to be the interval

$$[\phi_0 + \theta/2, \phi_0 - \theta/2],$$

where θ is the field of view of the camera and ϕ_0 is the viewing direction of the sensor.

A point q is considered to lie *within angular limits* of the point p_0 if the following condition holds

$$(\phi_0 + \theta/2) \geq \arctan ((y - y_0) / (x - x_0)) \geq (\phi_0 - \theta/2)$$

and *outside angular limits* otherwise.

A point q is *visible* from p_0 under the constraints of AL, the angular visibility limit, and R, the range of vision of the sensor, if it satisfies the following three conditions simultaneously :

(i) The line of sight constructed from p_0 through q does not intersect any polygon edges between points p_0 and q.

(ii) q is within range R of p_0.

(iii) q is within angular limits of p_0.

A line segment L_{i_j} is said to be visible if every point on the line segment is visible from p_0.

2.1.3. Finding Visible Obstacles in the Range of Vision

The problem is to find the set of visible line segments, defined by the set of polygons $\{P_i\}$, relative to some view $v_0 = (p_0, \phi_0)$. The range, R, and angular visibility limit, AL, define a region of visibility which has the shape of a sector of a circle. To simplify the analysis, the arc of the sector is approximated by a single straight line, making the visibility region a triangle. This is a good approximation only for size of angular visibility limit less than 90 degrees. This will not be a constraint in the simulation because any large angular visibility limit can be simulated by piecing together simulations for small angles at different orientations ϕ at the same point (x, y).

Two cases will arise in this problem :

(i) A polygon lies completely inside the visibility region (Fig. 2.1).

(ii) A polygon lies partly inside the visibility region (Figs. 2.2a and 2.2b).

In addition, a polygon may be occluded by another polygon inside the visibility region (Fig. 2.3).

The following algorithm is used to identify the visible line segments and their shadow regions:

(i) The set of polygons $\{P_i\}$ in the world is ordered by increasing distances of their centroids from p_0, the current location of the sensor.

(ii) The visibility region, called the visibility-range polygon (VRP), is initially approximated by the triangular region described previously.

(iii) The ordered set of polygons is clipped against the *clipping window* defined by the VRP, storing the portions of polygons which lie inside the VRP as a list of vertices in clockwise order. The clipping algorithm used is that of Sutherland-Hodgman [Foley82], which clips a convex or concave polygon against a convex window. The clipped polygons will have line segments which lie completely inside VRP. This simplifies the problem of finding visible line segments. For polygons which have no portion lying inside VRP, the fact is noted.

A simplification in this step is possible but has not been implemented. A parameter specifying the largest distance of any vertex from the centroid of each polygon can be calculated. If the distance of the centroid from the location of the sensor minus the value of this parameter is greater than the

range of vision of the sensor, the algorithm does not need to clip this polygon.

(iv) We define the *angular sweep* of a polygon P_i by the minimum and maximum angles its vertices make with the x-axis of a Cartesian plane with the location of the sensor as origin. Call these angles $\theta_{\min}$ and $\theta_{\max}$ respectively and call the associated vertices of the polygon $V_{i_{\min}}$ and $V_{i_{\max}}$. Starting at the beginning of the set of ordered polygons, if a portion of a polygon lies in VRP, the angular sweep of that portion is calculated. The case of a polygon lying across the positive x-axis has to be treated specially since angles are measured clockwise. In this case, $V_{i_{\min}}$ will be the vertex with maximum angle and $V_{i_{\max}}$ will be the vertex with minimum angle. The algorithm also takes note of the fact that if two successive vertices V_{i_j} and $V_{i_{j+1}}$ of a polygon P_i make the same angle θ with the x-axis, then the line segment L_{i_j} defined by the vertices will overlap with the line of sight. This line segment is classified as not visible.

(v) Since the vertices of a polygon P_i are ordered clockwise in the data structure, line segments $V_{i_{\min}}V_{i_{\min+1}}$, $V_{i_{\min+1}}V_{i_{\min+2}}$ ... $V_{i_{max-1}}V_{i_{\max}}$ will be visible. This is true since polygons in our world by definition are convex. It is also assumed that the height parameter associated with each polygon is large enough to put the remaining line segments of the polygon P_i inside the shadow cast by the visible line segments. Thus the visibility of all the line segments of a polygon need not be checked. Figure 2.4 illustrates this fact.

2.1.4. Finding the Occluded Regions in the Range of Vision

The height parameter, h_i, of each polygon which is specified arbitrarily is used to calculate the shadow cast by the visible line segments of the polygon. The lines of sight of the vertices $V_{i_{min}}$ and $V_{i_{max}}$ are extended away from p_0, the location of the sensor, by a distance h_i to determine their respective shadow points, tracing two new line segments. The two shadow points are joined by a third line segment. These three new line segments along with the visible line segments of the polygon form the *occluded* region of the polygon P_i with respect to the present view, v_0. Figure 2.4 illustrates this.

The net result of the simulation up to this point is to identify all obstacle regions (given by visible line segments) and the occluded regions. The remainder of the VRP is colored as free.

The case that has not been considered explicitly until now is identifying visible line segments of a polygon occluded by another polygon inside the VRP as illustrated by Figure 2.1. This is taken care of as follows. The whole VRP is first colored *white* (free region). The coloring of the occluded region and visible line segments as obstacles in that order is done for the *ordered* set of polygons, i.e. the nearest polygon to p_0 is treated first. A visible line segment is colored as an obstacle if and only if the previous color of the pixel is *white*, that is, it was in a free region. It is not colored *black* (obstacle region) if the previous color was *grey* (occluded region). So, if part of a visible (obstacle) line segment of the next nearest polygon lies inside the occluded region of a previous polygon, it would not

be colored as an obstacle.

2.1.5. Updating the Map of the Environment

As the mobile robot moves and takes successive scans from its sensors, it acquires new knowledge about its environment. This knowledge must be consolidated with the existing world model to produce an updated map of the world.

The vision sensor simulation does this as follows: Updating needs to be done for only those points in the map which are currently in an occluded or an unexplored region because these will be the regions that will get exposed as free space or obstacles. If a point in the map is currently in an unexplored region, it is given the color of the same point in the output of the current simulation. If a point in the map is in an occluded region, it is also given the color of the same point in the output of the current simulation only if the color is not that of an unexplored region. The updated map is used for path planning and obstacle avoidance.

3. PATH PLANNING PROCESS

The task of using the world model to find an unobstructed path to a selected goal is the path planning process. Since there may be many alternative routes to a goal, a path planning algorithm introduces a measure of path cost to define a selection criteria for optimal path search. A path planning algorithm also eases the task of navigating by representing the world map by certain primitives suitable for navigation.

Conventional path planning algorithms can be divided into two categories depending on the primitives they use for representing the world. In the first category are the methods which make trivial changes to the world map before planning a path. The regular grid search and vertex graph methods fall into this category. The methods in the second category make elaborate representation changes to convert the world map to a representation which is easier to analyze before planning a path. Free space methods, medial axis transform methods and Voronoi methods fall into this category. As pointed out in [Kambh86] mobile robots need a representation which is a compromise between these two categories. [Kambh86] has described path planning algorithms based on a hierarchical (region quadtree based) representation of the world map. The path planning cost for quadtree based search is lower compared to methods in the first category and the representation overhead involved is less than that of methods in the second category. Although the path produced by this algorithm is not strictly optimal, it is "negotiable" and it can be computed quickly. Other advantages of this representation is that the path can be easily constrained to satisfy conditions like

minimal clearance of the path and that the search can be staged, reducing the planning cost substantially.

We have used this algorithm for the path planning process. The cost metric used is the distance measured along the path. The robot will maintain continuous motion by alternatively asking for a goal from a higher-level navigation module and planning a path to it. At the time of planning the robot has access to all the information represented in the map plus features observable from the robot's current location. From the robot's viewpoint, parts of the world may be obscured by occlusion and by distance. In trying to reach the selected goal which lies in the robot's current view, the path planning process can regard all the occluded regions as obstacles and look for a path. Or the robot can try to map parts of its environment along the path in hope of discovering a shorter path to the current goal. This is what we call the optimistic path planner and is described below.

3.1. Optimistic Path Planner

This section describes a path planner which plans a locally optimal path from a start point to a given goal location in the robot's world consisting of free regions, obstacles, occluded regions and unexplored regions. Staged path planning with a region quadtree representation of the robot's world [Kambh86] is used to plan a negotiable path with occluded and unexplored regions assumed to be obstacles. The aim of this algorithm is to develop a negotiable path quickly rather than an "optimal path" , which is a costly operation. Our objective is to deter-

mine the impact of revealing occluded regions in the robot's world by sensing at a point along this path, and to decide whether replanning would be cost-effective for this path or not. We call the algorithm the *optimistic path planner*. The optimistic path planner will reveal occluded regions in the vicinity of the planned path as it traverses the path in the hope of discovering a lower cost path to the goal.

The cost of a path can be measured in terms of the energy expended by the mobile robot in traversing that path. It will depend, among other things on two factors : the length of the path and the number of direction changes in the path. The smaller the distance traveled and the fewer the direction changes made by the robot, the lesser will be the time and energy used by the robot in traversing the path. Hence, a straight line path between the start and goal points would be the optimal path. The goal is to reveal occluded regions in the vicinity of the negotiable path to develop a path which is closer to a straight line path.

3.1.1. Acceptable Paths

We define a path as *acceptable* if no replanning is required for it. Let l_1 be the length of a path planned under the assumption that all occluded regions in the world are actually obstacles, and let l_2 be the length of the path planned under the assumption that all occluded regions are actually free regions.

The ratio l_2/l_1 is a measure of the impact of the occluded space on the negotiable path. If the ratio is very close to 1, then revealing the occluded regions would not help in finding a lower cost path. But calculating the ratio l_2/l_1 will

require running the local path planner twice. To avoid this extra computation, consider the length l_3 of the straight line path from the start point to the goal point in question. The straight line path is the least expensive path that can be traversed by the robot to reach the goal point. The straight line path between the start point and the goal point may not be navigable but the ratio l_3/l_1 provides a heuristic measure of acceptability of the path. This ratio is always less than or equal to 1. If l_3/l_1 is very close to 1, the path P is very close to the straight line path between the start and goal points. Replanning by revealing the occluded regions in the vicinity of the path P will not be worth the cost. The path P is acceptable. If l_3/l_1 is close to zero, revealing occluded regions in the vicinity of the path P may help in finding a lower cost path.

3.1.2. Decision Procedure

Once it has been decided that the path P given by the local path planner is not acceptable, a decision has to be made concerning from which point on P to use the vision system. P will be a sequence of points $\{p_1, p_2, ..., p_n\}$ specifying the quadtree nodes constituting the path. The decision procedure will depend on two factors : (1) the amount of occluded space that can be potentially revealed from a particular point on the path, and (2) the distance to be traveled to reach that path point.

We want to take the next image from a point where a large amount of occluded space is revealed and small distance is traveled to reach it. Let $V(p_i)$ denote the area of occluded space visible from the path point p_i. The decision

can be based on a gain function computed for each path point, which will be directly proportional to the visibility measure of a path point, $V(p_i)$ and inversely proportional to the distance traveled, $S(p_i)$. Our experiments have shown that a gain function of the form

$$q(p_i) = V(p_i) \,/\, S(p_i) \tag{1}$$

tends to give more weight to path points with small $S(p_i)$, giving path points very close to the start point as best candidate points. Our current decision procedure analyzes the shape of the derivative of the visibility measure, $V(p_i)$, with respect to distance traveled along the path, at the discrete set of path points given by the coordinates of the quadtree nodes constituting the path. Thresholding this derivative at some appropriate value will give the path points at which the visibility is increasing at a sufficient rate. Starting at the first path point, we calculate the visibility measure for successive path points until we find a path point whose visibility derivative is above threshold. We continue with the calculation for successive path points while the derivative is above the threshold. The last path point for which the derivative is above the threshold is the point at which we take the next image of the world to reveal occluded regions.

3.1.3. Computing the Visibility Measure

An estimate of visibility, i.e., the number of pixels of occluded space that may get revealed from p_i, can be computed by analyzing the line of sight from p_i to each of the constituent points of occluded space in the robot's map of the

environment. If, for any such point, this line of sight does not intersect an obstacle or another occluded region, then we assume that this point of occluded space is visible from p_i. This computation is done under the assumption that occluded space hides a free region.

This analysis can be done in either of the two representation schema of the world we have available : the region quadtree representation of the world, which supports the path planning algorithms, or the raster representation, which is the output of the sensor system, i.e., vision simulation.

In the region quadtree representation, a line of sight between two points can be traced by finding all the leaf nodes of the quadtree that lie on the straight line joining these two points. One of the ways this can be accomplished is by a top-down traversal of the region quadtree. A node of the quadtree represents a square region of the picture, and it carries information about the size, color and location of this region in the image. The color information of the quadtree node identifies the type of region - free, obstacle, occluded or unexplored. A leaf node of a quadtree is a tip node of the tree. It represents a uniformly colored square region of the picture. A gray node of the quadtree is a node which is not a leaf node. It represents a square region which is a mixture of free, obstacle, occluded and unexplored space.

To traverse the quadtree top-down, we first visit the four sons of the root node. For each son, if the line intersects it and it is a gray node, we visit its sons and repeat the process recursively until we determine all the leaf nodes which intersect the line of sight. It can be shown that the average number of nodes, leaf

or gray, checked for intersection with the line of sight in this process is bounded by $4n$, where n is the number of leaf nodes lying on the line of sight. This result is proved in the Appendix.

The sequence in which the leaf nodes are identified by this process is not necessarily ordered according to their location on the line of sight. However, it is necessary to determine the correct ordering of these nodes. The line of sight is constructed between a path node and a node representing some occluded region, say R. A region in the image will be represented by a set of leaf nodes in the quadtree which are in the same connected component. Our goal, recall, is to determine whether the line of sight passes through an obstacle region or an occluded region which is other than R. On one hand, the top-down traversal of the region quadtree can be terminated when a leaf node is found which lies on the line of sight and represents an obstacle region. On the other hand, if a leaf node found in this process represents an occluded region it has to be determined whether this leaf node is in the connected component of the region R. One way would be to do the connected component labeling in the quadtree. Another way would be to determine the sequence of nodes through which the line of sight passes. The latter can be done simultaneously with the top-down traversal of the quadtree, as follows.

A leaf node of the quadtree which lies on the line of sight intercepts a segment of that line. Except for the special case of the horizontal or vertical line of sight, both the x and y coordinates of the endpoints of the intercepted segments are monotonic along the line of sight. As the leaf nodes come out of the top-down

```
Procedure LINE_OF_SIGHT (B,p1,p2);

                /* B is the gray node representing a region in the image
                   and p1 and p2 are the endpoints of the line of sight to be
                   traced in the image. The algorithm computes the leaf nodes
                   of the quadtree lying on the line of sight and sorts them
                   according to their relative locations on the line of sight */

begin
     ∀ node ∈ Son (B) ;
     if (node is gray) and (intersect (node, p1, p2) = true)
     then
        LINE_OF_SIGHT (node, p1, p2);
     else
          if (node is leaf) and (intersect (node, p1, p2) = true)
          then
             sort(node);
end LINE_OF_SIGHT;
```

Table 3.1. Line of Sight Algorithm

traversal of the quadtree, they can be sorted using a binary tree by their relative location on the line of sight, given by the x or y coordinate of one of the endpoints of the intercepted segment. The color of each node visited by the inorder traversal of this binary tree will give the sequence of type of regions the line of sight passes through. The detailed procedure for tracing a line of sight in the region quadtree of two-dimensional space is given in an algorithmic fashion in Table 3.1. This algorithm for tracing a line of sight in a region quadtree was not included in our implementation instead the line of sight was traced in the raster

representation of the path planning world.

In the raster representation, a line of sight can be traced between two points by computing the coordinates of the pixels which lie near the line. A number of algorithms exist in the literature for scan-converting lines which give different speed versus image quality trade-off. Since image quality was of little consideration to us we based our selection on speed and simplicity of implementation. We used the "Basic Incremental Algorithm" from [Foley82].

The visibility measure attributed to each path point or path node will be the total number of pixels of occluded space which are visible from the path point, or the area of quadtree nodes representing occluded space which are visible from a path node, depending on the method used for calculation.

In simple cases of path planning, we can assume that we never have to backtrack, which might be required if we have trap-like obstacles in the world. Hence the visibility measure of a path point can be computed only for those constituents of occluded regions which lie within a 180^0 field of view of the path point under consideration, and viewing in the direction of the goal. Also, the constituents of occluded regions considered for this purpose must be within the range of the sensor. Figure 3.1 illustrates this.

To determine whether a point lies in the appropriate 180^0 field of view of a path point, we consider the projection of the vector going from the path point to the point under consideration on the vector traced from the path point to the goal point. If the value of the projection calculated is positive then the point lies

in the field of view of the path point.

3.1.4. Optimistic Path Planner Algorithm

The following steps outline the implementation of the algorithm used for the optimistic path planner. The flow of the algorithm implementation is shown in Figure 3.2. The input is a two-dimensional model of the path planning world with specified shapes and locations of obstacles. A vision simulator is used to decompose this path planning world into free, obstacle, occluded or unexplored regions.

(i) The initial position of the robot and the first goal are chosen.

(ii) Vision is simulated from the current robot position with a particular orientation of the sensor. This orientation will be specified by some high-level module in the navigation system based on global path planning constraints. The sensor orientation does not affect the optimistic path planner algorithm. The output of this step is a binary array or raster representation of the robot's environment decomposed into free regions, obstacles, occluded regions and unexplored regions.

(iii) Under the assumption that the occluded and unexplored regions are obstacles, the staged path planning algorithm is run, using a quadtree representation of the path planning world, to obtain a negotiable path from the present position of the robot to the goal. The output of this step is a path, P, consisting of a sequence of quadtree nodes $\{p_1, p_2, \ldots, p_n\}$ from the start to the goal point.

(iv) The ratio l_3/l_1 for the path P is evaluated. If this ratio is above a specified threshold value, i.e the path is acceptable, we ask for the next goal after traversing the acceptable path and go to step (ii) to repeat the process, or we stop if we have reached the final goal point.

(v) If P is not acceptable, the visibility analysis is performed on the path and the path point is determined from which to reveal occluded regions by taking new images.

(vi) The path is traversed until this path point is reached.

(vii) The vision simulator is run for a 180^0 field of view with the sensor aligned in the direction of the goal, revealing some occluded regions. The output of this step is again a raster representation of the robot's environment decomposed into free regions, obstacles, occluded regions and unexplored regions.

(viii) Using a quadtree representation of the output of step (vii), the staged path planning algorithm is run again to obtain a negotiable path from the present position to the goal. We go back to step (iv) to evaluate the path.

3.2. Results

Experiments were performed on a 64×64 pixel image map containing 27 obstacles. With a vision sensor range of 50 pixels, the vision sensor simulation took typically 2 seconds to run on a VAX - 11/785 running 4.3BSD Unix with ten obstacles in its range of vision. The simulator program was written in C.

Figures 3.4a-e show the result of running the optimistic path planner on a sequence of paths returned by the staged path planning algorithm of [Kambh86]. In the figures, free space is shown by white area, obstacles are black areas, occluded regions are indicated by light cross-hatching and unexplored regions by dots. The path is shown by dark cross-hatching and the start and goal nodes in the quadtree by single-hatching. Below each figure is given the value of the ratio l_3/l_1 which is used to determine whether the path requires replanning or not.

In Figures 3.4a and 3.4c, the path is found to be unacceptable because the value of the ratio is below an acceptable threshold. The value of this threshold was set at 0.60 after performing a large number of experiments on improving negotiable paths in this particular obstacle configuration. The path is analyzed to find the best candidate path point from which more images of the world are taken in the direction of the goal point. New improved paths are shown in Figures 3.4b and 3.4d. The values of the ratios l_3/l_1 for the new paths are found to lie above the threshold, hence no more improvement of the paths is attempted.

These results show that substantial improvement can be obtained in the path by running the optimistic path planner. The sequence of goals to be

attained successively to reach the final goal point lie in the sensor range in each view and are selected at random. In a complete system, goals would be given to the path planner by some high-level navigation module.

4. CONCLUSIONS

In this report we have shown how to treat occluded regions in a mobile robot's path planning space. We demonstrated a procedure for deciding when (or whether) to employ the sensor system during execution of a negotiable path to, potentially, reveal the occluded regions as obstacles or free space for the purpose of replanning.

Short range dynamic path planning for a mobile robot presents many interesting problems because of robot's incomplete and limited knowledge of its environment. The problem of reaching a goal point will involve generation of a sequence of subgoals which will have to be successively attained to reach the final goal point. It would involve selecting an "optimal" subgoal from among candidate subgoals at each step. Another problem presented is "representation" of occluded and unexplored regions in the robot's path planning space. As occluded and unexplored regions are mapped, more information about them becomes available and the representation of the robot's environment should be able to incorporate such changes. We should be able to make changes in the representation of the image map with relatively low cost.

A major difficulty in dynamic path planning stems from the fact that the robot perceives its environment through sensors and keeps track of its position through mechanisms that are inevitably imprecise. It is clear that these imprecisions will lead to world model degradation, and identification of objects and areas already known and modeled would be, in general, impossible. This difficulty will arise when the optimistic path planner tries to reveal occluded regions in the

vicinity of the path by taking more images of the environment. To solve this problem, the error in the information about objects and robot position should be modeled for the sensors concerned. Landmark navigation, i.e., referencing by fixed and known beacons, will allow the system to reduce the robot's positional uncertainity below some upper bound. Knowing the sensory uncertainities, the positions of the perceived terrain features, and given the locations of previously detected features, the new perspective may be matched against the old by varying the estimated vehicle position (within error bounds) until the best fit is found.

Unexplored regions, i.e., the areas outside the robot's field of view, can also be mapped in trying to find the shortest possible path to the goal. Again, a decision procedure will have to be employed to find out whether it is worthwhile looking at the unexplored regions in the vicinity of the current path. The two-dimensional world model used for path planning may be enhanced to categorize areas by slope, texture, altitude, etc. This may influence the cost of a path. Other measures which can also be included in building a cost metric for the path planning process are the time and energy required to traverse a path. Also, previously executed paths may be remembered in case the region has to be traversed again.

APPENDIX

Theorem 1

Given a quartered block B and a sample of lines distributed uniformly in space and orientation which pass through the block B. Then the average number of quadrants intersected by a line passing through the block is 2. [Nelson87]

Theorem 2

The average number of nodes checked for intersection by LINE_OF_SIGHT is bounded by 4n, where n is the number of leaf nodes in the quadtree which lie on the line of sight.

Proof

From Theorem 1, the total number of siblings to be checked for intersection at the level of leaf nodes which lie on the line of sight on an average is 2n.

Going up one level in the quadtree, the average number of gray nodes which are parents of these siblings is 2n / 4.

On an average, the number of siblings to be checked for intersection at this level is 2. 2n / 4.

And so on until we reach the root node.

Summing up, the average number of nodes checked for intersection by LINE_OF_SIGHT is bounded by

$$2.n + 2.\frac{2n}{4} + 2.\frac{1}{4}.(\,2.\frac{2n}{4}\,) + \ldots$$

$$= 2.n\,(\,1 + \frac{1}{2} + \frac{1}{4} + \ldots)$$

$$< 2.n\,(\,2\,) = 4.n$$

Acknowledgements : We have benefited greatly from discussions with Randal Nelson.

REFERENCES

[Brooks85]
Brooks, R.A., "Visual Map Making for a Mobile Robot", *IEEE Conference on Robotics and Automation*, St. Louis, MO, March 1985, 824-829.

[Davis85]
Davis, L.S., Andresen, F., Eastman, R., Kambhampati, S., "Visual Algorithms for Autonomous Navigation", *IEEE International Conference on Robotics and Automation*, St. Louis, MO, March 1985, 856-861.

[Foley82]
Foley, J.D. and Van Dam, A. "*Fundamentals of Interactive Computer Graphics*", Addison-Wesley, 1982.

[Kambh86]
Kambhampati, S. and Davis, L.S., "Multiresolution Path Planning for Mobile Robots", *IEEE Journal of Robotics and Automation* **2,** 1986, 135-145.

[Koch85]
Koch, E. *et al.*, "Simulation of Path Planning for a System with Vision and Map Updating", *IEEE Conference on Robotics and Automation*, St. Louis, MO, March 1985, 146-160.

[Kuan85]
Kuan, D.T., Zamiska, J.C., Brooks, R.A., "Natural Decomposition of Free Space for Path Planning", *IEEE Conference on Robotics and Automation*, St. Louis, MO, March 1985, 168-173.

[Lozano79]
Lozano-Pérez, T. and Wesley, M.A., "An Algorithm for Planning Collision Free Paths among Polyhedral Obstacles", *Communications of the ACM* **22,** 1979, 560-570.

[Lozano81]
Lozano-Pérez, T., "Automatic Planning of Manipulator Transfer Movements", *IEEE Transactions on Systems, Man, and Cybernetics* **11,** 1981, 681-698.

[Meystel84]
Meystel, A. and Koch, E., "Computation Simulation of Autonomous Vehicle Navigation", *Proc SPIE Vol 485, Applications of Artificial Intelligence*, 1984, 159-168.

[Nelson87]
Nelson, R., "Population Analysis of Quadtrees with Variable Node Size", University of Maryland Center for Automation Research Technical Report in preparation, 1987.

[Pavlidis82]
Pavlidis, T., "*Algorithms for Graphics and Image Processing*", Computer Science Press, 1982.

[Samet82]
Samet, H., "Neighbor Finding Techniques for Images Represented by Quadtrees", *Computer Graphics and Image Processing* **18,** 1982, 37-57.

[Samet84]
Samet, H., "The Quadtree and Related Hierarchical Data Structures", *ACM Computing Surveys* **16,** 1984, 187-260.

[Thompson78]
Thompson, A.M., "The Navigation System of the JPL Robot", *International Joint Conference on Artificial Intelligence*, 1978, 749-757.

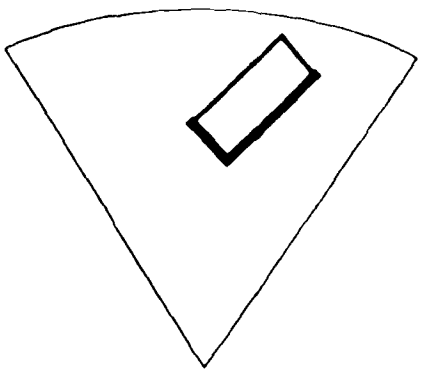


Figure 2.1 Obstacle lies completely inside the visibility region.

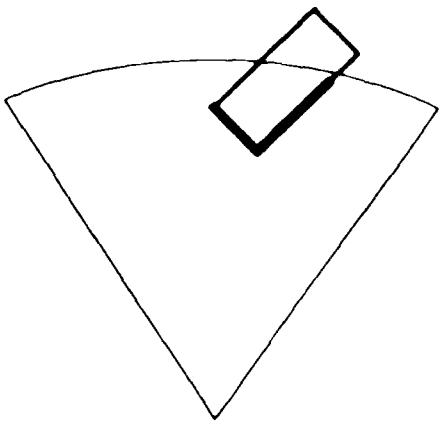


Figure 2.2a Obstacle lies partly inside the visi ity region (limited by the range lir it).

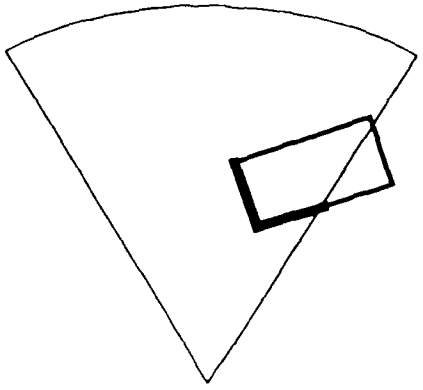


Figure 2.2b Obstacle lies partly inside the visibil- ity region (limited by the angular visibility limit)

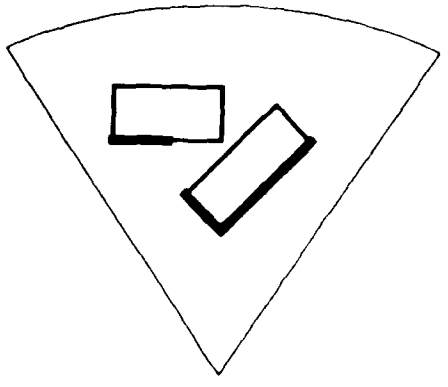


Figure 2.3 Obstacle is occluded by another o tacle inside the visibility region

In all these cases, the visible line segments of obstacles are shown by thick lines.

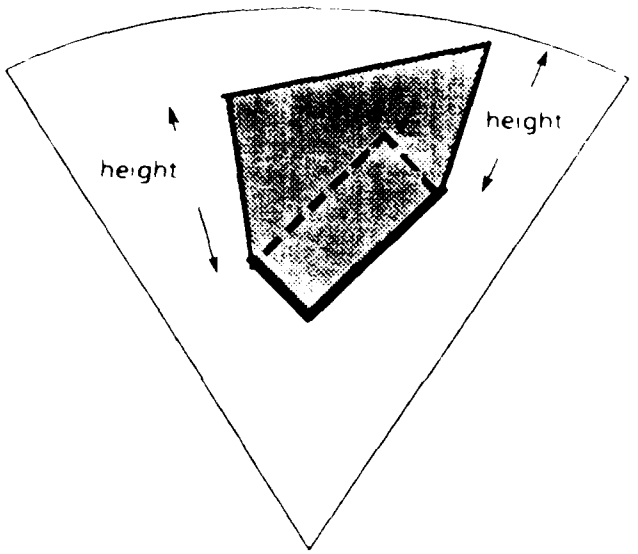


Figure 2.4 Outlining the occluded region of an obstacle by using its height specification

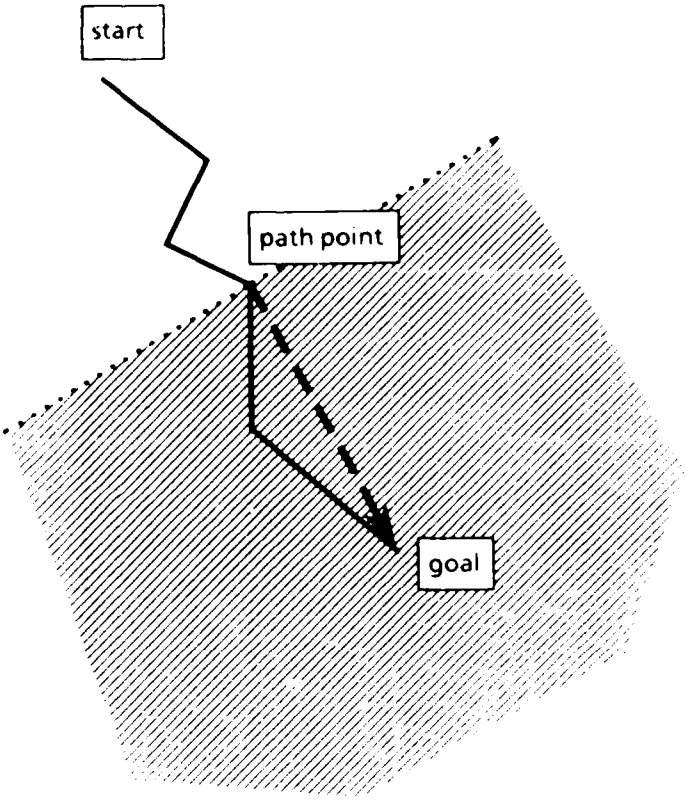


Figure 3.1 The visibility measure for path points is computed from occluded regions within a 180^0 field of view looking in the direction of the goal. The shaded region in the above diagram shows the field of view considered for a path point and the broken arrow shows the direction of view.

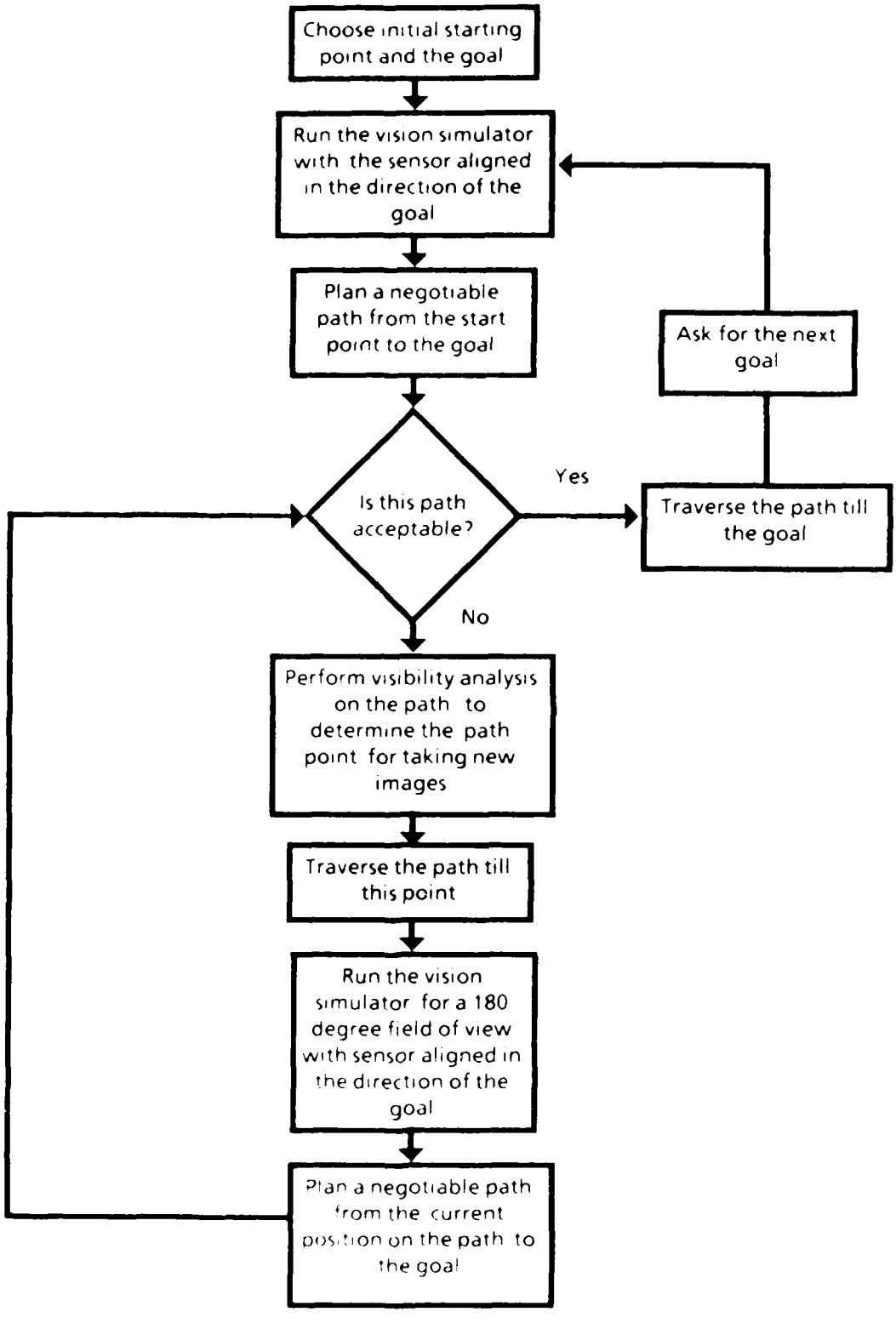


Figure 4.2 Implementation of the Optimistic Path Planner Algorithm

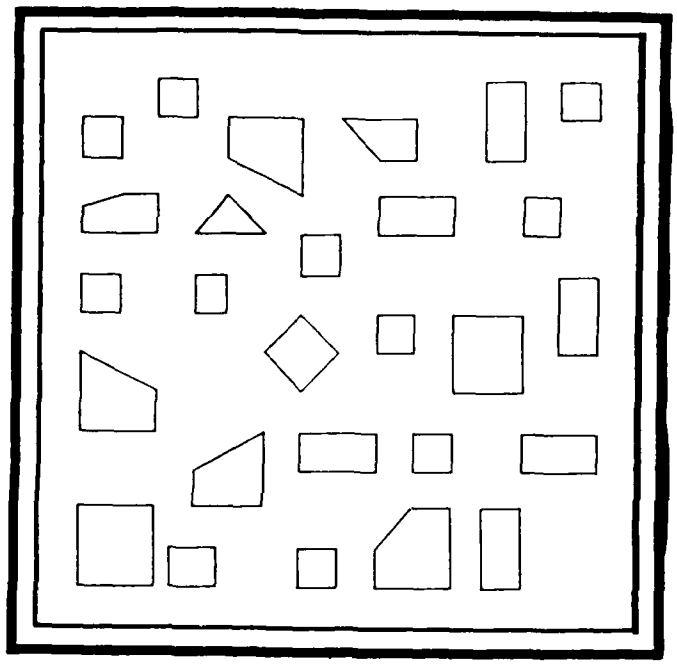
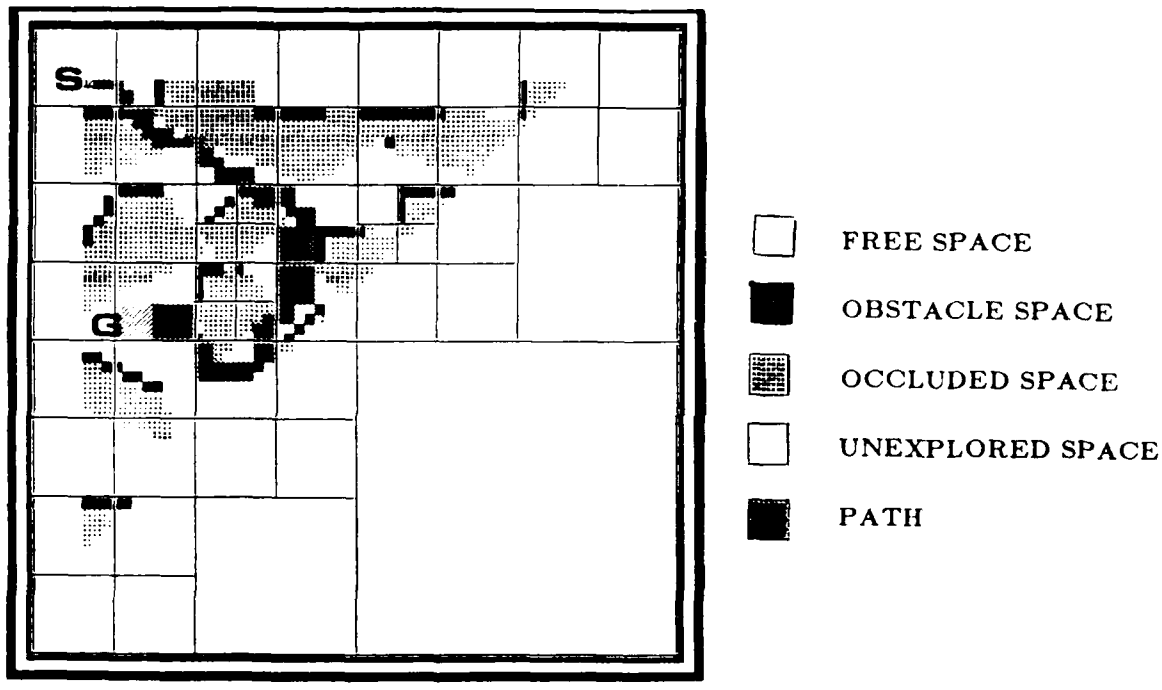


Figure 3.3 This figure shows the shapes and locations of obstacles defining the world used for testing the optimistic path planner algorithm.



$L_3/L_1 : 0.36$

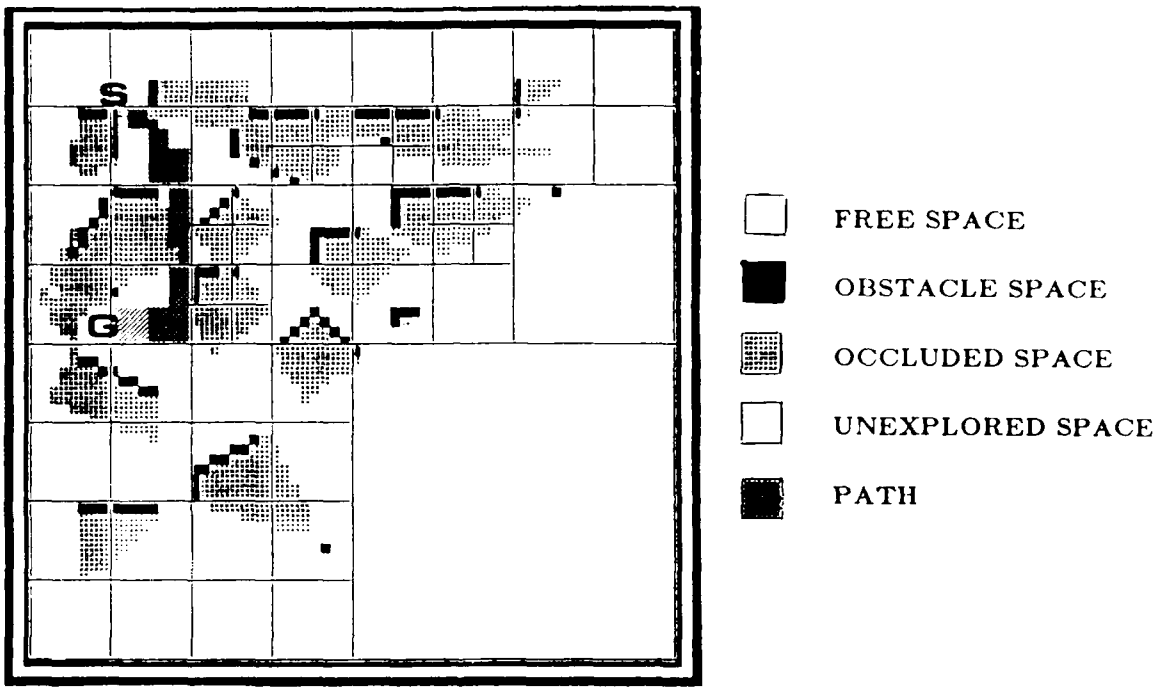
Figure 3.4a Given a start point and a goal point, a negotiable path is planned assuming that all the occluded space consists of obstacles. Since the value of the ratio l_2/l_1, the measure of acceptability of a path, is below a threshold, the path is declared as unacceptable and an attempt is made to improve the path by using the optimistic path planner algorithm.

x	y	v	dv	thres_dv *	dist
6	5	75	3	0	2.00
6	7	82	1	0	4.00
6	9	84	60	1	6.00
8	9	205	-7	0	8.00

* threshold for dv = 4

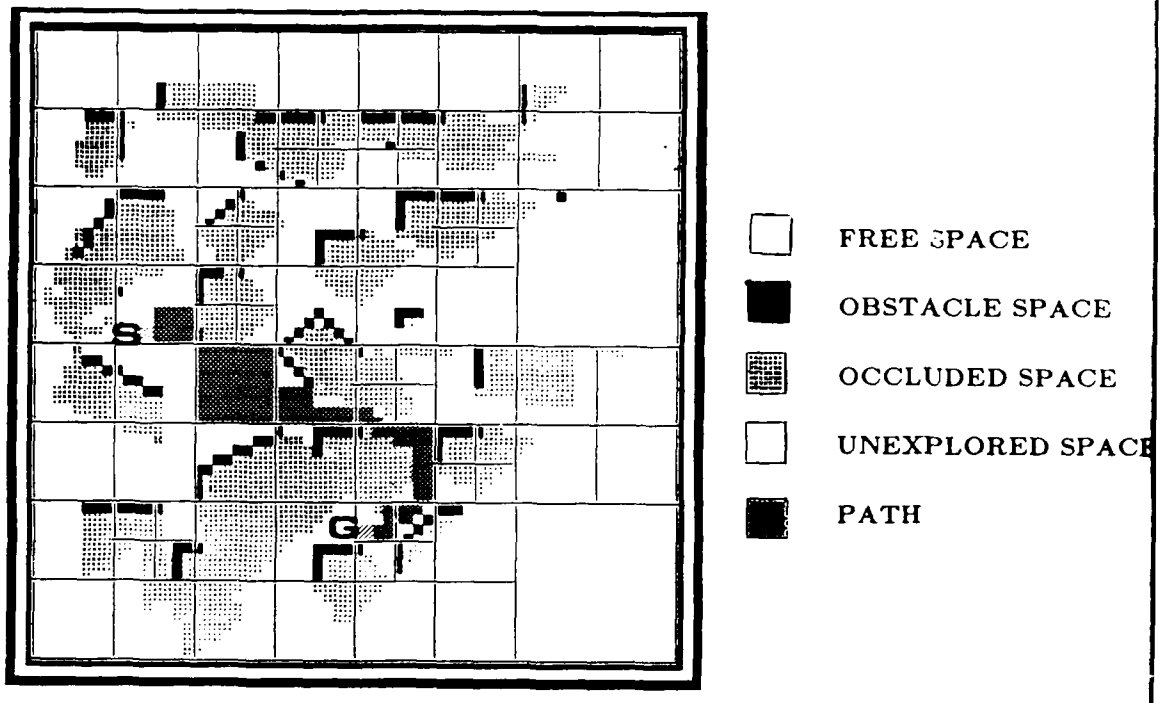
x, y	Coordinates of a quadtree node on the path
v	Amount of occluded space visible from (x,y)
dv	Derivative of v with respect to distance traveled along the path
thres-dv	Thresholded value of dv
dist	Distance traveled along the path

Analysis of the negotiable path shown in Figure 3.4a indicates that new images should be taken from the path node with coordinates (6,9). The total cpu time taken by this analysis was 2.0 seconds.



L_3/L_1 : **0.81**

Figure 3.4b After traversing the path to the point determined by the optimistic path planner algorithm, new images of the environment are taken. A negotiable path is again planned from this point to the goal. This path is determined as acceptable.



L_3/L_1 : 0.57

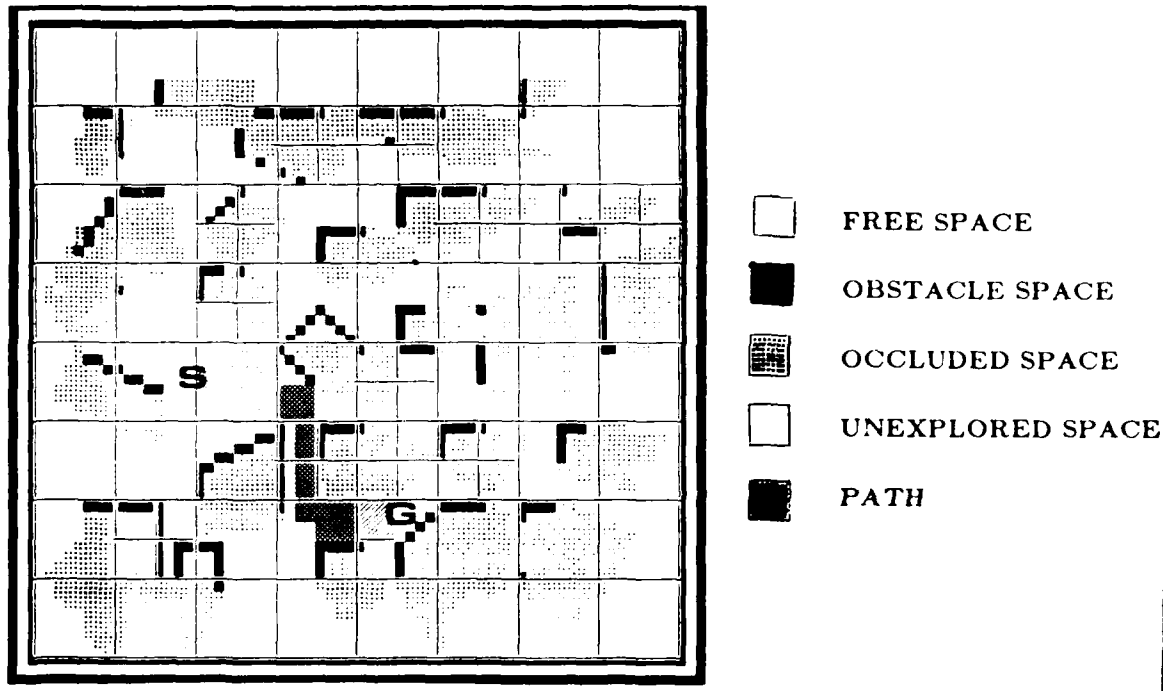
Figure 3.4c A negotiable path is planned from the current position to the next goal point. The value of the ratio l_3/l_1 is found to be below an acceptable threshold and an attempt is made to improve the path by using the optimistic path planner algorithm

x	y	v	dv	thres_dv *	dist
29	14	187	-1	0	3.16
30	17	182	-18	0	6.32
35	20	73	17	1	12.16
37	26	184	-26	0	18.48

* threshold for dv = 4

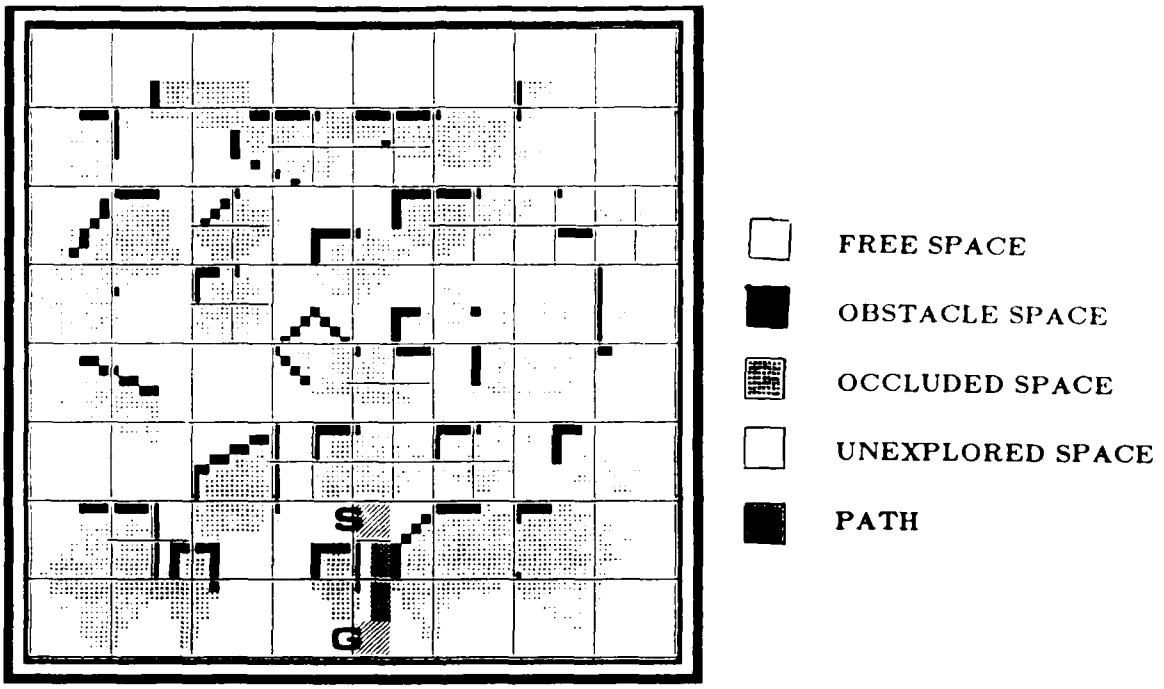
x, y	Coordinates of a quadtree node on the path
v	Amount of occluded space visible from (x,y)
dv	Derivative of v with respect to distance traveled along the path
thres-dv	Thresholded value of dv
dist	Distance traveled along the path

Analysis of the negotiable path shown in Figure 3.4c indicates that new images should be taken from the path node with coordinates (35,20). The total cpu time taken by this analysis was 2.0 seconds.



L_3/L_1 : 0.81

Figure 3.4d After traversing the path to the point determined by the optimistic path planner algorithm, new images of the environment are taken. A negotiable path is again planned from this point to the goal. This path is determined as acceptable, and hence no replanning is done for it.



L_3/L_1 : 0.90

Figure 3.4e A negotiable path is planned from the current position to the next goal point. No attempt is made to improve this path as the value of the ratio l_3/l_1, the measure of acceptability of a path, is above the threshold.

END

10-87

DTIC