A 'RETRACTION' METHOD FOR TERRAIN MODEL ACQUISITION

Nageswara S. V. Rao
Department of Computer Science
Louisiana State University
Baton Rouge, LA 70803

N. Stoltzfus
Department of Mathematics
University of Michigan
Ann Arbor, MI 48105

S.S. Iyengar
Department of Computer Science
Louisiana State University
Baton Rouge, LA 70803

ABSTRACT

We consider the following problem (called *the terrain model acquisition problem*): A point robot R is placed in a finite-sized two-dimensional obstacle terrain populated by a set $O = \{O_1, O_2, \cdots, O_n\}$ of unknown polygonal obstacles. Each obstacle O_i is a finite-sized polygon with a finite number of vertices. Initially the number of obstacles in the terrain is unknown to R. And also, the number and the locations of vertices of each obstacle are unknown to R. The robot is equipped with sensors that detect all vertices and edges that are visible from the present location of the robot. The robot is required to navigate and acquire the *complete* terrain model in a finite amount of time. In this paper we propose a solution based on the *retraction* method, and this method has the advantage of keeping the robot as far as possible from the obstacles during the navigation. We also present a method for terrain model acquisition by a circular robot R of radius r, $(r > 0)$.

Keywords and Phrases:

Voronoi diagram, retraction, terrain model acquisition

1. INTRODUCTION

In recent years, there has been a large amount of research reported in literature concerning algorithms for robot navigation. Some of the most pioneering works for navigation planning in *known* terrains (i.e. the terrain models are precisely known) are due to Lozano-perez and Wesley [3], Reif [9], Schwartz and Sharir [10], and O'Dunlaing and Yap [5]. These problems, often referred to as *find-path* problems, deal with planning a collision-free (possibly optimal) navigational path for a robot through a terrain cluttered with obstacles of 'known' locations and configurations. Of late there has been growing interest in algorithms for robot navigation in *unexplored* or *unknown* terrains (i.e. the terrain model is not known *a priori*). In such situations the path planning is based on the sensor information. Lumelsky and Stepanov [4] present a sound theoretical framework for these algorithms for a point robot operating amidst simple close-curved obstacles.

In the area of navigation in unexplored terrains, here we deal with a specific problem wherein a robot is required to autonomously acquire the complete terrain model. The main motivation for this problem comes from the fact that once the terrain model is known, one can apply find-path algorithms for computing a navigation path (possibly optimal according to a chosen measure) to any required destination point. Then the robot can move along the planned path without sensor usage. Moreover, the sensor-based navigation algorithms occasionally lead the robot into local detours due to the localized nature of the sensor information. Such detours could be avoided by acquiring the terrain model first and employing find-path algorithms for planning paths to the destination points.

We define the *terrain model acquisition problem* as follows: A point robot R is placed at a point in a finite-sized two-dimensional terrain populated by an unknown (but, finite) number of simple polygonal obstacles. Each obstacle is of finite size (i.e. there exists a circle of finite radius that encloses the obstacle) and is formed by an unknown (but finite) number of vertices. The robot is equipped with a sensor system that detects all vertices and edges that are visible from the present location of the robot. The robot is equipped with a navigational system that moves it along a specified straight line or a second order curve. It takes a finite amount of time to traverse a finite amount of distance either along a straight line or along a second order curve. The robot is required to autonomously navigate in the terrain and build the *complete* obstacle terrain model and return to the starting point in a finite amount of time.

In the sequel, we use the following notation. We denote the set of obstacles by $O = \{O_1, O_2, \cdots, O_n\}$, where O_i is a finite-sized polygon in plane with a finite number of vertices. $VER(O_i)$ denotes the set of vertices of O_i. $N = \sum_{i=1}^{n} |VER(O_i)|$ denotes the total number of obstacle vertices. The free-space is denoted by Ω $(= O_1^C \cap O_2^C \cap \cdots \cap O_n^C)$. The convex hull formed by the vertices of a set of polygons $\{P_1, P_2, \cdots, P_k\}$ is denoted by $C(\{P_1, P_2, \cdots, P_k\})$. The detection of all vertices and edges seen from a point using the sensors is termed as a single *scan* operation.

In this paper we present a solution to this problem based on the retraction method proposed by O'Dunlaing and Yap [5]. The robot incrementally constructs a graph, called the *navigational course* based on the Voronoi diagram of the terrain from the sensor information that is acquired time-to-time. The navigational course which is the union of a subset of the Voronoi diagram and an envelop (to be defined formally) of the terrain. Since the terrain is unknown, this navigational course is not available to R to start with. The R locally constructs the edges

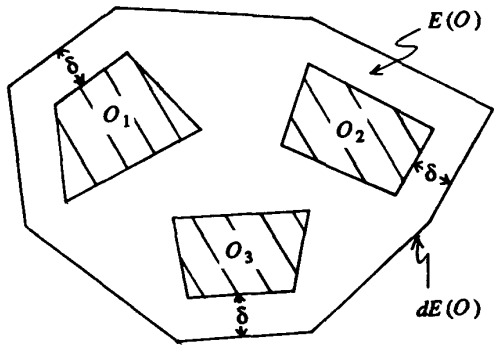

Fig.1. $E(O)$ of the terrain of O_1, O_2, and O_3.

and vertices of the navigational course from each sensor operation (scan operation). Then R moves to a computed vertex along the computed edges. R keeps navigating on the navigation course (R backtracks at some points) till it acquires the complete terrain model; at this point it moves back to the starting point. The correctness of the algorithm follows from the connectedness of the navigational path. We then show that our method is directly applicable to a circular robot of radius r (>0).

We then compare the proposed solution with the visibility graph based methods discussed in Rao et al [6,7]. The former has the advantage of minimizing the collision probability by navigating the robot 'away' from the obstacles. The storage complexity of the proposed method is $O(N)$ as opposed to $O(N^2)$ of the latter. Moreover, the present method has a overall computational complexity of $O(N^2 \log N)$ whereas the latter has $O(N^3)$. Both the methods require $O(N)$ sensor operations. Finally the extension of visibility graph method to a finite-sized (circular) robot calls for significant extensions to the basic algorithm (as shown in [7]). The proposed method is directly applicable to a circular robot.

The organization of the paper is as follows: In section 2, we present the terrain acquisition algorithm and its correctness is established in section 3. We analyze the performance of the algorithm in section 4. In section 5, we present a method for circular robot of radius r, ($r>0$). In section 6, we compare our method with the visibility graph based approaches. In this paper we present the basic results without proofs (see Rao et al [8] for details about the proofs).

2. TERRAIN ACQUISITION ALGORITHM

We first present a definition of the navigational course, and then describe our terrain acquisition algorithm. For any $x \in \Omega$, define $Near(x)$ as the set of points that belong to the boundaries of obstacles O_i, $i=1,2,...,n$ and are closest to x. The *Voronoi diagram*, $Vor(O)$, of the terrain populated by O is the set of points:

$\{ x \in \Omega \mid Near(x)$ contains more than one point $\}$

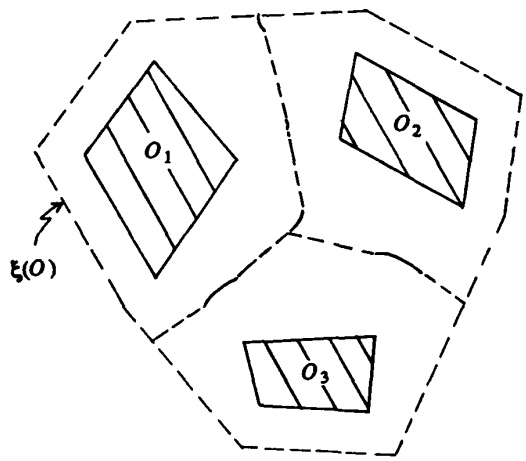

Fig. 2. $\xi(O)$ of the terrain O_1, O_2, and O_3.

In this case, $Vor(O)$ is a union of $O(n)$ straight lines and parabolic arcs (see Kirkpatrick [2] for more details on the Voronoi digrams). Each of this line or parabolic arc is referred to as a *V-edge*. The points at which the edges meet are called *V-vertices*. Furthermore, $Vor(O)$ can be specified as a combinatorial graph in which each edge is labeled with two end V-vertices, and an equation defining it as a curve in the plane. Each V-vertex is labeled with its coordinates.

Consider the convex hull $C(O)$ of union of vertices of all obstacles (i.e. convex hull of $\bigcup_{i=1}^{n} VER(O_i)$). Let $E(O)$ denote the region that is contained within an envelop of clearance of δ around $C(O)$ as shown in Fig.1. Let us define *navigational course* for O as $\xi(O)=(Vor(\Omega) \cap E(O)) \cup dE(O)$, where $dE(O)$ is the boundary of the envelop $E(O)$. In fact $\xi(O)$ precisely contains the Voronoi diagram of O that lies inside $E(O)$ and the boundary of $E(O)$. See Fig.2. for an example. Note that the set of vertices of $\xi(O)$ is the union of V-vertices, vertices of the envelop $E(O)$ and the intersection points of $Vor(O)$ and $\partial E(O)$. Similarly the edges of $\xi(O)$ is the union of edges of $Vor(O)$ and $E(O)$. The vertices (edges) of $\xi(O)$ are henceforth referred to as ξ-*vertices* (ξ-*edges*). It is easy to see $\xi(O)$ as a planar graph formed by ξ-vertices and ξ-edges. The set of all ξ-vertices that are adjacent to a ξ-vertex v constitute the set of *neighbors* of v.

We now present the terrain model acquisition algorithm. We treat $\xi(O)$ as a combinatorial graph made up of ξ-vertices and ξ-edges. Initially the point robot R is located at an arbitrary point in Ω at a finite distance from an obstacle. Then R performs a scan and computes the vertices of $\xi(O)$. If R lies outside $C(O)$ (i.e. all obstacles are found to be to one side of a line through the location of R) then R computes a vertex v_0 of $E(O)$ and moves to it. If R lies inside $C(O)$, then R computes the Voronoi diagram of visible part of the terrain (say using Kirkpatrick's [2]) and moves to a V-vertex v_0. From v_0, the

```
algorithm VACQUIRE(v);
begin
1.  perform scan operation;
2.  construct the ξ(O) of the visibility region;
3.  update the partially constructed ξ(O);
4.  mark all concave corners as visited;
5.  if (v has unvisited neighbors)
6.  then
7.     move to a nearest unvisited neighbor v_1;
8.     VACQUIRE(v_1);
9.  else
10.    backtrack (computationally) on the path v from v_0
       and find v_b with unvisited neighbors;
11.    if (such v_b exists)
12.    then
13.       find v_2 the nearest unvisited neighbor of v_b;
14.       compute a shortest path to v_2 on the available ξ(O);
15.       move to v_2 along the computed path;
16.       VACQUIRE(v_2);
17.    else
18.       move to v_0 along shortest path;
19.       move to the starting point;
end;
```

algorithm VACQUIRE is invoked, which moves R from one ξ-vertex to the other. The basic idea of VACQUIRE is as follows: R navigates entirely on the 1-skeleton $\xi(O)$; at each ξ-vertex v a scan is performed, the neighbors of v are computed, and the $\xi(O)$ is updated. The R keeps visiting the new ξ-vertices till it encounters a ξ-vertex with all visited neighbors. At this point R 'backtracks'. This process is continued till all the ξ-vertices are visited. At this point R moves back to v_0 and then to the starting point.

The detailed algorithm is given in algorithm VACQUIRE. At some intermediate stage, let R be located at ξ-vertex v (initially $v=v_0$). Now, R performs a scan operation (line 1). If all detected vertices lie only to one side of a line through v, (R is outside $C(O)$) then it computes the adjacent vertex of $E(O)$. If not, the Voronoi digram of the visible part is computed (using, say Kirkpatrick's [2]) (line 2). The partially built $\xi(O)$ is updated by entering the adjacency list of v (line 3). Note that only the ξ-vertices which are adjacent to v are stored in the partial model of $\xi(O)$. All ξ-vertices that correspond to concave corners of obstacles are marked as visited (line 4). Then if v has unvisited neighbors then R moves to a nearest of them (line 5-7), and then VACQUIRE is recursively invoked (line 8).

If all neighbors of v are visited then the path from v_0 to v is accessed. This path is stored on a stack. Whenever a new ξ-vertex is visited it is pushed onto the stack i.e. the ξ-vertices are pushed onto stack as R visits them. The stack is accessed when all neighbors of v are visited. At this point, the top element of the stack is repeatedly popped till a vertex v_b with at least one unvisited neighbor is found. At this point the stack contains a path from v_0 to v_b. This process corresponds to line 10. If such v_b exists then it's nearest unvisited neighbor v_2 is found and a path to v_2 is computed using single source shortest path algorithm of Frederickson [1] for planar graphs (This algorithm has a time complexity of $O(M\sqrt{\log M})$ as opposed to $O(M^2)$ algorithm of Dijkstra for a planar graph of M nodes. The former exploits the planarity of the graph to lower time complexity.) (line 14). Note that this path to v_2 is shortest on the available portion of $\xi(O)$ and in general not the shortest in the entire $\xi(O)$. Then R moves to v_2, and VACQUIRE is recursively invoked from v_2 (lines 15-16). If no such v_b exists, then R moves to v_0 along the shortest path (line 17-18). Then R moves back to the starting point (line 19) at which VACQUIRE terminates. we discuss the performance of this algorithm subsequently in this paper.

3. CORRECTNESS OF THE ALGORITHM

We first show that all ξ-vertices will be visited by R, and a scan is performed from each ξ-vertex at the termination of VACQUIRE.

Lemma 1: *The 1-skeleton $\xi(O)$ is topologically connected. Hence, the combinatorial graph corresponding to $\xi(O)$ is (graph) connected, i.e., there exists a path consisting of ξ-edges between any two ξ-vertices.* □

A close look at the algorithm VACQUIRE reveals that the execution of VACQUIRE is equivalent to carrying out a 'conceptual' depth-first-search on the graph of $\xi(O)$ assuming it is available. Hence, we have the following Lemma.

Lemma 2: *The order in which new ξ-vertices of graph of $\xi(O)$ are visited is same as that of carrying out a depth-first-search on the graph of $\xi(O)$.* □

Using a cellular decomposition of Ω, we show the following lemma [8].

Lemma 3: *Every point in $\Omega \cap E(O)$ is seen during a scan operation from some ξ-vertex.* □

We need to show that the $\xi(O)$ is constructed correctly from the scan (visibility) information. Note that from any ξ-vertex v, only the ξ-vertices that are adjacent to v are updated. Consider the cellular decomposition of the terrain based on ξ-edges as shown in Fig.3. As shown in Fig.3 (a),(b) and (c), each straight line ξ-edge adjacent to v contains two convex regions (cells) - one to each side -, and this entire region is seen from v. If ξ-edge is parabolic then the cell can be decomposed into a triangle and a cone (Fig.3(d)), and the entire region is seen from either of the ends of the given ξ-edge. It is clear that if the computed points lie on the boundary $dE(O)$ of the envelop then the computed vertices exactly correspond to the actual vertices of the envelop. Consider the case where the computed vertices contain V-vertices. This part of the Voronoi diagram contains the points which are nearest to edges and vertices seen from v. By the separability notion discussed by Kirkpatrick [2] this part of the computed diagram corresponds the actual Voronoi diagram. Thus the navigational course will be correctly constructed by the algorithm VACQUIRE. We

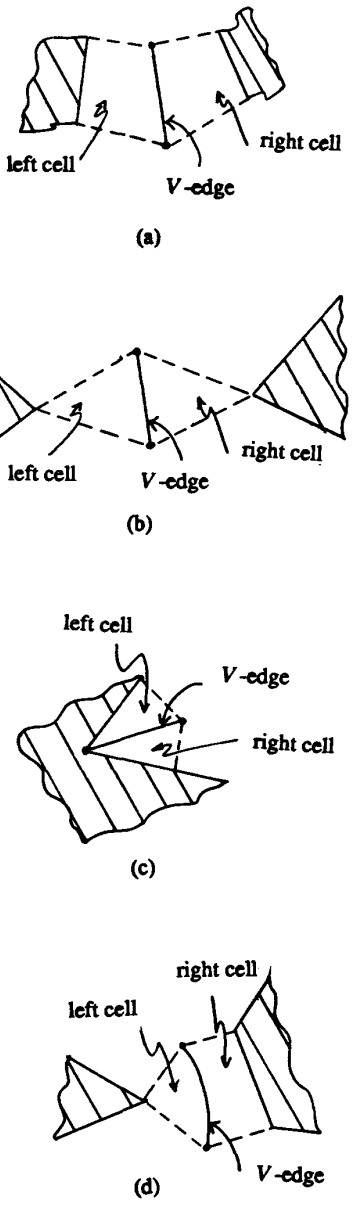

Fig.3. Cellular decomposition of Ω based on edges.

prove the correctness of the algorithm VACQUIRE in the following theorem.

Theorem 1: *The algorithm VACQUIRE acquires the complete obstacle terrain model in finite amount of time.*

Proof: In the next section we show that $\xi(O)$ is a finite graph and the distance traversed by R is bounded by a finite number (Lemma 4). From Lemmas 1 and 2 it follows that R visits all ξ-vertices, and scans from them in finite amount of time (since a depth-first-search on a connected graph, with finite number of node, visits all vertices in finite amount of time). Now from Lemma 3, all vertices and edges of all obstacles are detected from some scan operation. The terrain model can be directly constructed from the information about the obstacle vertices and edges. Or, the terrain model can be reconstructed from $\xi(O)$ by storing the distance information with each ξ-edge. □

In the next section we evaluate the complexities of scan operations, distance traversed and computations of the algorithm VACQUIRE executed on the robot R.

4. PERFORMANCE

We now present some properties of $\xi(O)$. Using a dual of $Vor(O)$, we estimate the bounds on the number of ξ-edges and ξ-vertices in the following Lemma (see [8] for details).

Lemma 4:

(i) $(n+5)/2 \le \#\xi\text{-vertices} \le 4N-n-2$

(ii) $3(n+1)/2 \le \#\xi\text{-edges} \le 6N-3n-3$ □

We use this lemma in estimating the complexity of sensor operations of VACQUIRE. We also estimate a bound on the distance traversed by the robot R executing VACQUIRE.

Theorem 2: *(i) The number of scan operations performed by robot while executing VACQUIRE is at most* $4N-n-2$, *(ii) The total distance traversed by the robot while executing VACQUIRE is at most twice the total length of the depth-first tree of* $\xi(O)$ *rooted at* v_0. □

We estimated the parameters such as number of scanning operations and total distance traversed by R while executing VACQUIRE. In the following theorem we estimate the complexity of computational activities carried by VACQUIRE. In our implementation we use the adjacency list representation of $\xi(O)$. We store the coordinates of each ξ-vertex in the adjacency lists. We maintain a table called MAP-TABLE. The table gives the visited information of a ξ-vertex specified by it's coordinates. The MAP-TABLE is implemented as an AVL-tree. One can store the information of these tables in the adjacency list. Then the complexity of finding whether a ξ-node (specified by its coordinates) visited or not is $O(N)$. The cost of this operation is $O(\log N)$ using the table.

Theorem 3: *The computational complexities of various tasks carried out by VCAUIRE are as follows: (i) the storage complexity is* $O(N)$, *(ii) cost of construction of* $\xi(O)$ *is* $O(N^2 \log N)$, *(iii) total cost of path planning is* $O(N^2\sqrt{\log N})$, *(iv) the cost of construction of MAP-TABLE is* $O(N \log N)$, *and the total cost of accesses to MAP-TABLE is* $O(N \log N)$. □

In the next section we compare the performance of the proposed terrain acquisition algorithm with vertex-based ones using the estimates developed in this section.

5. CIRCULAR ROBOT

Now consider a circular robot R of radius r (>0). For each $x \in \Omega$, we define *clearance* (x) to be the distance to a nearest member of $Near(x)$. We then consider $E(O)$ such that the celarance $\delta = r$. We now define a modified navigational course $\xi^*(O)$ as follows:

$$\xi^*(O) = \{x \in \xi(O) \mid clearance(x) \ge r\}$$

Let R be located at x_0 (i.e. the center of R is located at x_0). For $x \in \Omega$, the clearance strictly increases along the line segment joining x to $\mathrm{Im}(x)$ [5]. Now, $\mathrm{Im}(x_0)$ will be contained in a maximal connected component F of $\xi^*(O)$. The component F contains ξ-vertices and ξ-edges with clearence greater than or equal to r. Some edges of F could be truncated versions of corresponding ξ-edges. In such cases we imagine a vertex at the truncated ends of the edges. Now this F plays the role of the navigational course for the algorithm VACQUIRE.

Let x be a point reachable by R from x_0. Then x can be 'seen' by first moving from x_0 to $\mathrm{Im}(x_0)$ along a straight line, and then moving to $\mathrm{Im}(x)$ along F. By using the arguments similar to those in Theorem 1, we can show that R will acquire all the obstacle vertices and edges that can be 'seen' by R (we use F in place of $\xi(O)$). The performance of this algorithm is upper-bounded by the estimates in Theorem 2 and Theorem 3.

6. COMPARISON WITH VERTEX BASED METHODS

In this section we compare our method with the visibility graph based methods of Rao et al [6,7]. The main motivation for our method comes from the implementation stand point. In general it is very difficult to navigate a robot arbitrarily close to the obstacle edges as is required by the visibility graph methods. This is a serious problem if robot works in an open loop mode based on some type of visibility sensor. This problem is present in retraction based methods if the terrains are densely populated, for example, if two obstacles are arbitrarily close to each other. However this is less problematic in a general case if retraction methods are used. The approach presented here is limited to two-dimensional terrains where as the method of [6] works for three dimensional terrains also. An extension to a finite-sized circular robot (of diameter δ) in terrains with a minimum clearance of δ around each obstacle boundary for two-dimensional terrains is presented in [7]. In such a terrain the clearance around each ξ-edge will be at least δ, and thus the algorithm presented here can be directly applied to that case.

In terms of the complexity of scan operations both the algorithms have the same $O(N)$ complexity. However, the scan operations are time consuming in a real-life implementation hence it is instructive to compare them more closely. The number of scan operations in the visibility graph methods is N always, where as the retraction method requires at most $4N-n-2$ and at least $(n+5)/2$ scan operations. The former requires that the robot be capable of straight line motion, where as the latter requires that the robot navigate along the second order curves (which can be implemented with straight line motion by using some polygonal path approximation to the second order curve). The distance traversed in the former case can be shown to be less than twice the total length of the depth-first tree on the visibility graph rooted at the starting vertex. In the later case the bound is twice the total length of the depth-first tree on the retract rooted at the starting vertex.

Table 1: Comparison of two methods

quantity for comparison	visibility graph method	retraction method
Storage	$O(N^2)$	$O(N)$
construction	$O(N^2)$	$O(N^2 \log N)$
path planning	$O(N^3)$	$O(N^2\sqrt{\log N})$
MAP-TABLE access	$O(N^2 \log N)$	$O(N \log N)$

We now look at the computational aspects. The visibility graph can have $O(N^2)$ edges and hence the storage complexity in this method is $O(N^2)$. The complexity of the retraction based method is $O(N)$ as a result of the planarity of $\xi(O)$. The complexity of the path planning in the former is $O(N^3)$ as opposed to $O(N^2\sqrt{\log N})$ in the latter. The cost of construction of the visibility graph is $O(N^2)$ where as that of the retract is $O(N^2 \log N)$. The construction cost of the MAP-TABLE is same in both, where as the access cost in the former is $O(N^2 \log N)$ as opposed to $O(N \log N)$ in the latter. This comparison is summarized in Table.1

7. CONCLUSIONS

In this paper, we have presented an algorithm that enables a point robot to acquire the complete model of a finite-sized two-dimensional obstacle terrain populated by an unknown (but finite) number of simple polygonal obstacles. We have shown that the terrain model will be completely built in a finite amount of time. We analyzed the algorithm for complexities of various operations. We then compared the performance of this algorithm with the existing algorithms based on visibility graph approaches. Future extension of this work deals with a finite-sized robot. Then the orientation problem becomes very significant. For a circular robot with a diameter δ, we can show (using the connectivity of the retract and the depth-first nature of the algorithm) that terrain model of all regions that are theoretically accessible to the robot will be acquired by the robot.

REFERENCES

[1] FREDERICKSON, G.N., Shortest path problems in planar graphs, Proc. 24th Ann. Symp. on Found. of Comput. Sci., 1983, pp. 242-247.

[2] KIRKPATRICK, D.G., Efficient computation of continuous skeletons, Proc. 20th Ann. Symp. on Found. of Comput. Sci., 1979, pp. 18-27.

[3] LOZANO-PEREZ. T. and M. WESLEY, An algorithm for planning collision-free paths among polyhedral obstacles, *Commun. ACM*, vol. 22, 1979, pp. 560-570.

[4] LUMELSKY, V.J. and A.A. STEPANOV, Dynamic path planning for a mobile automation with limited information on the environment, *IEEE Trans. on Automatic Control*, vol. AC-31, no.1, 1986, 1058-1063.

[5] O'DUNLAING. C. and C.K. YAP, A "retraction" method for planning the motion of a disc, *J. algorithms*, vol.6, 1985, pp. 104-111.

[6] RAO, N.S.V., S.S. IYENGAR, J.B. OOMMEN and R.L. KASHYAP, Terrain acquisition by a point robot amidst polyhedral obstacles, Proc. 3rd IEEE Conf. on AI Appl., Orlando, Fl., Feb. 1987, pp. 170-175 (a refined version to appear in *IEEE J. Robotics and Automation*).

[7] RAO, N.S.V., S.S. IYENGAR, C.C. JORGENSEN and C.R. WEISBIN, On terrain acquisition bya finite-sized mobile robot in plane, Proc. 1987 IEEE Int. Conf. Robotics and Automation, Raleigh, NC.,1987, pp. 1314-1319.

[8] RAO, N.S.V., N. STOLTZFUS and S.S. IYENGAR, The terrain acquisition by mobile robots: IV. A 'retraction' method, Tech. Rep. #87-015, Dep. Computer Sci., Louisiana State University, 1987.

[9] REIF, J., Complexity of mover's problems and generalizations, Proc. 20th Ann. Sympo. Found. Comput. Sci., 1979, pp.421-427.

[10] SCHWARTZ, J.T. and M. SHARIR, On the piano-movers problem: I. The case of two dimensional rigid body moving amidst polygonal barriers, *Comm. Pure Appl. Math.*, vol.36, 1983, pp. 345-398.