Because of the number of setups directly affects the fixturing, tolerancing, and machining time, an algorithm has been developed to determine the minimum number of setups required for producing the part. The algorithm first attempts to fathom the possibility of achieving the target part from a single setup. This step involves exploring the possibility of achieving the target part from a single machining direction in the set of candidate machining directions defined by the enclosing object faces. If multiple setups is inevitably required, then the algorithm searches from the minimum number of setups to achieve the target part by enumeration.

As an illustration, consider the part shown in Fig. 14(a). This part is machinable from the enclosing object MEB since all the part faces are visible. The system then determines that at least two machining setups are required to achieve the finished part requirement. The "removal volume" formed by the union of valid prisms and residues from each setup is subsequently constructed as shown in Fig. 14(b). The prisms provide depth information for material removal from the part in the form of $z = f(x, y)$ where z is the depth from the enclosing object (or raw stock) face, and (x, y) are the coordinates of a point on an enclosing object face. Using this depth information with linear interpolation techniques, cutting strategies and a tool path can be derived (see Fig. 14(c)). The generated enclosing object and the orientation of the enclosing object face can be used as a reference for the raw stock and fixture positioning, respectively.

V. Conclusions

We have presented a machinability test that determines whether a given part can be machined from a certain set of machining directions, such as those obtained from the convex hull, the minimum enclosing box, or by the augmentation of these surfaces. The analysis of machinability focuses only on polyhedral parts. However, curved surfaces can also be incorporated via polygonal faceting. This provides an automatic method for segmenting the surface into portions that need to be accessed from different directions.

Acknowledgment

The authors would like to thank Dr. G. L. Hogg and Dr. R. J. Mayer of Texas A&M University for their support of this research and A. A. Keen for his helpful discussions.

References

[1] G. T. Armstrong, G. C. Carey, and A. DePennington, "Numerical code generation from a geometric modeling system," in *Solid Modeling from Computers*, M. S. Pickett and J. Boyse, Eds. New York: Plenum, 1984.

[2] A. Parkinson, "The use of solid models in BUILD as a database for NC machining," in *Software for Discrete Manufacturing*, J. P. Crestin and J. F. McWaters, Eds. New York: Elsevier, 1986.

[3] D. S. Nau and T. C. Chang, "Prospects for process selection using artificial intelligence," *Computers Industry*, vol. 4, pp. 253-263, 1983.

[4] A. A. G. Requicha and J. Vandenbrande, "Automated systems for process planning and part programming," in *Artificial Intelligence Implications for CIM*, A. Kusiak, Ed. New York: Springer-Verlag, 1988.

[5] T. C. Chang and R. Wysk, "An integrated CAD/automated process planning system," *AIIE Trans.*, pp. 223-233, 1981.

[6] A. A. Requicha and H. B. Voelker, "Constructive solid geometry," Tech. Memo. 25, Production Automation Project, Univ. of Rochester, Rochester, NY, 1977.

[7] J. D. Foley and A. Van Dam, *Fundamentals of Interactive Computer Graphics*. Reading, MA: Addison-Wesley, 1982.

[8] S. C. Chan and H. B. Voelker, "An introduction to MPL: A new machining process/programming language," in *Proc. IEEE Int. Conf. Robotics Automat.* (San Francisco), Apr. 1986.

[9] F. Kimura, H. Suzuki, and L. Wingard, "A uniform approach to dimensioning and tolerancing in product modeling," in *Computer Applications in Production and Engineering*, K. Bo, L. Estensen, P. Falster, and E. A. Warman, Eds. New York: Elsevier, 1987.

[10] C. J. Su, "An enclosing object based automatic NC code generation system," Ph.D. dissertation, Dept. of Industrial Engineering, Texas A&M Univ., College Station, TX, 1989.

A "Retraction" Method for Learned Navigation in Unknown Terrains for a Circular Robot

Nageswara S. V. Rao, N. Stoltzfus, and S. S. Iyengar

***Abstract*—We consider the problem of learned navigation of a circular robot R, of radius δ (≥ 0), through a terrain whose model is not *a priori* known. W consider two-dimensional finite-sized terrains populated by an unknown (but, finite) number of simple polygonal obstacles. The number and locations of the vertices of each obstacle are unknown to R; R is equipped with a sensor system that detects all vertices and edges that are visible from its present location. In this context, we deal with to problems: the visit problem and the terrain model acquisition problem. In the visit problem, the robot is required to visit a sequence of destination points, and in the terrain model acquisition problem, the robot is required to acquire the complete model of the terrain. We present an algorithmic framework for solving these two problems based on a retraction of the free space onto the Voronoi diagram of the terrain. We then present algorithms to solve the visit problem and the terrain model acquisition problem.**

***Keywords*—Incidental learning, polygons, retraction, unknown terrains, Voronoi diagrams.**

Nomenclature

n	Number of obstacles.
p	Number of vertices of outer face of $D(O)$.
s	Expansion factor in obtaining $E(O)$ from $C(O)$.
C	Number of concave vertices.
O	Set of obstacles or terrain.
O_i	ith obstacle polygon.
M	Number of positions in the visit problem.
N	Total number of obstacle vertices.
R	Robot.
S_1	Set of visited nodes by NAV.
S_2	Set of nodes with neighbors visited by NAV.
δ	Radius of the robot.
Ω	Free-space.
Ψ	Maximal connected component of set of free positions of R.
Γ	Sum of Ψ and R.
$C(O)$	Convex hull of vertices of O.
$D(O)$	Dual graph of Vor (O).
$E(O)$	Expanded convex hull of O.

Manuscript received April 24, 1990; revised May 7, 1991. Preliminary results of the terrain model acquisition portion of this work were presented at the IEEE International Conference on Robotics and Automation, April 25-29, 1988, Philadelphia, PA.

N. S. V. Rao is with the Department of Computer Science, Old Dominion University, Norfolk, VA 23529-0162.

N. Stoltzfus is with the Department of Mathematics, Louisiana State University, Baton Rouge, LA 70903.

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

IEEE Log Number 9102214.

Vor (O)	Voronoi diagram of O.
Vor_1 (O)	Modified Voronoi diagram.
Vor_1^* (O)	Connected component of Vor_1 (O).
$\xi(O)$	Navigation course of terrain O.
$\partial E(O)$	Boundary of $E(O)$.

I. Introduction

The problem of planning collision-free paths for moving a body through a terrain whose model is known has been extensively studied under the popular generic appellation of the piano movers problem. Some of the most important contributions to the solution of this problem are due to Lozano-Perez and Wesley [7], O'Dunlaing *et al.* [11], Reif [17], and Schwartz and Sharir [18]. Sharir [19] and Yap [20] present excellent surveys of various solutions to this problem. In these problems, a precise model of the terrain is available for the purpose of path planning.

Another interesting problem of robot navigation deals with the collision-free navigation through an unknown terrain, i.e., a terrain whose model is not *a priori* known. Abelson and diSessa [1] presented an algorithm for a point robot to escape from planar mazes, using touch sensing. Lumelsky and Stepanov [8] presented algorithms for a point robot to move from a source point to a destination point, using touch sensing, in a planar terrain populated with arbitrary shaped obstacles. Cox and Yap [2] have recently developed algorithms to navigate a rod to a destination position in planar polygonal terrains (i.e., planar terrains populated by polygonal obstacles); here the rod has touch-sensing ability, and the solution is based on the notions of retraction.[1] For planar polygonal terrains with convex obstacles, Oommen *et al.* [12] have developed algorithms for a point robot to navigate to a destination point, and at the same time "learn" about the parts of terrain encountered on the way to the destination. Here, the robot uses a combination of touch sensing and distance probing. Earlier, Iyengar *et al.* [3] have presented a similar approach in the context of heuristic navigation algorithms. Rao *et al.* [13] and Rao and Iyengar [16] solved the same problem for a point robot equipped with a sensor that obtains all the visible obstacle boundaries; the obstacles need not be convex in this case. In the method of [12], the visibility graph of the terrain is incrementally constructed. It is shown in [16] that a subgraph of the visibility graph (obtained by removing the edges and vertices corresponding to concave obstacle vertices) suffices to ensure the correctness of the navigation algorithm. Also, the same method can be extended to the case of a circular robot [16]. The above problems can be grouped under the generic name of the visit problem, wherein a robot is required to visit a sequence of destination points through an unknown terrain.

Another important problem, called the terrain model acquisition problem, deals with autonomously building a terrain model by a robot. The case of a point robot in three-dimensional terrains populated by polyhedral obstacles has been studied by Rao *et al.* [13]; in this formulation the robot can detect all visible obstacle boundaries. This method is based on incremental construction of the visibility graph of the terrain. Lumelsky *et al.* [9] proposed two methods for acquiring the terrain model, in planar terrains populated arbitrary shaped obstacles, by a point robot equipped with a touch sensor.

In the case of robot sensor capable of detecting visible obstacle boundaries, Rao [15] showed that the visit problem and the terrain model acquisition problem are intimately related to each other in that both can be solved under a single framework. A graph, called the navigation course, that satisfies a set of required properties can be used as an underlying structure for any graph search algorithm; such approach provides algorithms to solve both the visit and terrain model acquisition problems [15]. A brief account of this framework is presented in Section II for completeness. The visibility-graph-based solutions of Oommen *et al.* [12], Rao *et al.* [13], [14], [16] for the visit problem, and of Rao *et al.* [14] for the terrain model acquisition problem can be unified under this framework (for these cases, the visibility graph is the navigation course). A retraction, from the free space onto the Voronoi diagram, yields another navigation course for a point robot, as stated in [15]. However, the treatment of [15] is tutorial in nature, and no technical details, such as proofs of the properties, etc., are given.

Research in these navigational problems has been motivated by a practical application involving the development of an autonomous robot intended for carrying out rescue operations in nuclear power plants in the events of radiation leaks [3]. A solution to the visit problem enables a robot to carry out a set of operations in different locations in unfamiliar environments. in this application, the motion planning is essentially sensor-based and involves expensive sensor operations. Furthermore, during the navigation, the robot may temporarily navigate into local detours because of the partial nature of information returned by sensors. By suitably "learning" a terrain model, one can reduce the number of sensor operations and also the number of detours [12]. Furthermore, if a complete terrain model is available, the robot can avoid local detours and also the expensive sensor operations. A dedicated rescue robot typically idles in between the rescue operations, which are fairly infrequent in most cases. In such situation, the robot could be employed to acquire a terrain model during the idle periods.

In this paper, we consider planar terrains composed of polygonal obstacles and a circular robot. We propose a retraction method, based on Voronoi diagram of the terrain, for solving both the visit problem and the terrain model acquisition. In this method, the robot mostly navigates along the Voronoi diagram of the terrain, and thus has an advantage of keeping as far away from the obstacles as possible. This aspect seems very important in practical implementations as the earlier methods, based on the visibility graphs, require navigation along obstacle boundaries. Additionally, the proposed method results in a storage complexity of $O(N)$ as opposed to $O(N^2)$ of visibility-graph-based methods, where N is the total number of obstacle vertices. Although the retraction method has been well-developed for known terrains, some additional work is needed to adapt it to unknown terrains. This work includes, among others, the introduction of additional vertices so as to ensure local constructibility property (to be defined later in Section III), obtaining tighter bounds on the number of sensing points, etc.

The organization of the paper is as follows: The basic framework of our solution is outlined in Section II; this section is a recapitulation of [15], and is provided for completeness. In Section III, we present the definition and properties of the navigation course to be used for navigational purposes. In Section IV, we present solutions to the visit problem and the terrain model acquisition problem.

II. Basic Algorithm

We consider a finite-sized two-dimensional terrain populated by a set $O = \{O_1, O_2, \cdots O_n\}$ (n is finite) of simple disjoint polygons, called the obstacles. Each obstacle O_i has a finite number of vertices. The terrain is completely unknown to R, i.e., the number of obstacles and the number of locations of vertices of each obstacle

[1] The term *retraction* corresponds to a continuous map from a topological space X to a subset A of X such that every point of A is mapped onto itself and every point in X-A is mapped onto some point in A [5]. Retractions were found to have properties that are conducive to efficient solutions for several motion planning problems in known terrains [20].

are unknown. The free space is given by

$$\Omega = \bigcap_{i=1}^{n} O_i^C$$

where O_i^C is the complement of O_i in the plane. The closure of the free space is denoted by $\bar{\Omega}$. Let N denote the total number of vertices of all obstacles and let C denote the total number of concave vertices.

Consider a circular body R of radius δ, $(\delta \geq 0)$; R is treated as an open disc of radius δ. R houses a computational device with a storage capability and is capable of moving along a straight-line path or a curved path of second degree (in either case the path is specified). The location of the center of R is called the position of R. Let x be a position of R. A point $y \in \bar{\Omega}$ is said to be visible to R if the straight line segment joining x and y is entirely contained in $\bar{\Omega}$. R is equipped with a sensor that detects the maximal set of points on the obstacle boundaries that are visible from its present location. Such an operation is called a scan operation. The visible vertices and edge intervals are listed in the clockwise direction. The scan operation is an abstraction of a 360° scan performed by a vision of a sonar system.

Initially, R is located at a point d_0 without intersecting any obstacle, and at a finite distance from some obstacle. In the terrain model acquisition problem [14]-[16], R is required to acquire the model of the terrain to a degree that it can navigate to any reachable destination (or conclude that the destination is not reachable) by using a path-planning algorithm of known terrains. Note that for a point robot this requirement is tantamount to acquiring the entire model of the terrain. In any case, no sensor operations are required for navigational purposes, after the terrain model has been completely acquired. In the visit problem, R is required to visit the positions $d_1, d_2, \cdots, d_M$ (in the specified order), if a path through these positions exists [12], [13]-[16]. If no such path exists, then R must report so within a finite amount of time.

We now present the algorithm NAV, which is the basic strategy used by R [15]. Here, R performs a "graph exploration type" of navigation using a combinatorial graph, called the navigation course, $\xi(O)$, of the terrain O. The nodes (edges) of $\xi(O)$ are called ξ nodes (ξ edges). $\xi(O)$ is a 1-skeleton embedded in Ω, i.e., each ξ vertex is a point in Ω and each ξ edge is a one-dimensional curve joining the vertices. Further, a ξ node specifies a collision-free position for R, i.e., a position for R such that it is entirely contained in Ω. An edge that joins two ξ nodes v_1 and v_2 specifies a collision-free path between v_1 and v_2. The $\xi(O)$ is initially unknown, and it is incrementally constructed using the data obtained through the sensor operations. The algorithm NAV is given below (a detailed explanation of NAV is given in [15]):

```
Algorithm NAV(v)
begin
1.  perform a scan operation from $v$;
2.  mark $v$ as visited, delete it from $S_2$ and append it to $S_1$;
3.  compute the adjacency list of $v$;
4.  append to $S_2$ all neighbors of $v$ that have not been visited;
5.  if ($S_2$ is not empty)
6.  then
7.      select $v^* \in S_2$;
8.      plan a path from $v$ to $v^*$;
9.      move to $v^*$;
10.     $NAV(v^*)$;
11. else
12.     return to start vertex $v_0$;
    endif
end
```

NAV is initiated with a ξ vertex $v = v_0$ and $S_2 = \{v_0\}$. The set S_1 contains all ξ vertices that are visited by R. The set S_2 contains all ξ vertices that are not visited by R, but each $v \in S_2$ is adjacent to some ξ vertex in S_1. There exists a path along the computed edges from any vertex of S_1 to any vertex of S_2. In each step, R, located at v, selects a $v^* \in S_2$, and then moves to v^*.

Suppose that $\xi(O)$ satisfies the property of local-constructibility, i.e., the adjacency list of a ξ vertex v can be computed from the information obtained by a scan operation performed from v. Further, suppose that $\xi(O)$ satisfies finiteness property, i.e., $\xi(O)$ has a finite number of vertices and edges. Also, let $\xi(O)$ satisfy graph connectivity property, i.e., any two ξ vertices are connected by a path of ξ edges. Then we have the following observation.

Observation 1: If $\xi(O)$ satisfies the properties of finiteness, connectivity and local-constructibility, then, R, executing the algorithm NAV, visits all vertices of $\xi(O)$ in a finite amount of time. ■

III. The Navigational Course

In this section, we present a specific navigation course $\xi(O)$ based on the Voronoi diagram of the terrain and prove some properties that are relevant to the navigation algorithms of the next section. In Subsection III-A, we present a preliminary structure Vor_1 (O), which is then modified in subsection III-B to yield the required navigation course $\xi(O)$.

A. Basic Structure

For $x \in \Omega$, we define Near (x) as the set of points such that each point belongs to the boundary of some obstacle O_i, $i = 1, 2, \cdots, n$, and is closest to x. The Voronoi diagram, Vor (O), of the terrain populated by O is the set $\{x \in \bar{\Omega} \mid \text{Near}(x) \text{ contains more than one point}\}$. In this case, Vor (O) is a union of $O(N)$ straight lines and parabolic arcs [4], [6], [21]. Each of these lines or parabolic arcs is referred to as a V edge. The points at which the edges meet are called V vertices. Fig. 1 shows an example of a Voronoi diagram for a simple terrain.

Consider the convex hull $C(O)$ of the union of vertices of all obstacles (Fig. 2(a)). Let $E(O)$ denote the polygonal region obtained by pushing the edges of $C(O)$ outward by a finite distance of $s \geq \delta$ and taking the interior of "grown" region as shown in Fig. 2(b). The exact value of s is inconsequential to the correctness of the present method, but it decides how far away the robot moves from the terrain during the navigation; in particular $s = \delta$ is an appropriate choice. We define Vor_1 (O) = (Vor $(\Omega) \cap E(O)) \cup \partial E(O)$, where $\partial E(O)$ is the boundary of $E(O)$. See Fig. 3 for an example. The set of vertices of Vor_1 (O) is the union of: 1) V vertices, 2) vertices of the envelope $E(O)$, and 3) intersection points of edges of $\partial E(O)$ with V edges. The set of edges of Vor_1 (O) is the union of: 1) edges of $\partial E(O)$, 2) V edges that are completely inside $E(O)$, and 3) V edges such that precisely one vertex lies inside $E(O)$ (in this case we only take the portion of the V edge that lies inside $E(O)$). The edges (vertices) of Vor_1 (O) are called V_1 edges (V_1 vertices). Vor_1 (O) is a planar graph formed by V_1 vertices and V_1 edges. The set of all V_1 vertices that are adjacent to a V_1 vertex v constitute the set of neighbors of v.

In the remainder of this section we investigate four basic properties of Vor_1 (O): 1) finiteness, 2) connectivity, 3) local constructibility, and 4) terrain visibility. Note that the first three properties have been discussed in the last section, and terrain visibility will be subsequently discussed in this section. These four properties together will be used to prove the correctness of our navigation algorithm. It can be shown that Vor (O) will also satisfy these four properties, but in the present application it would require

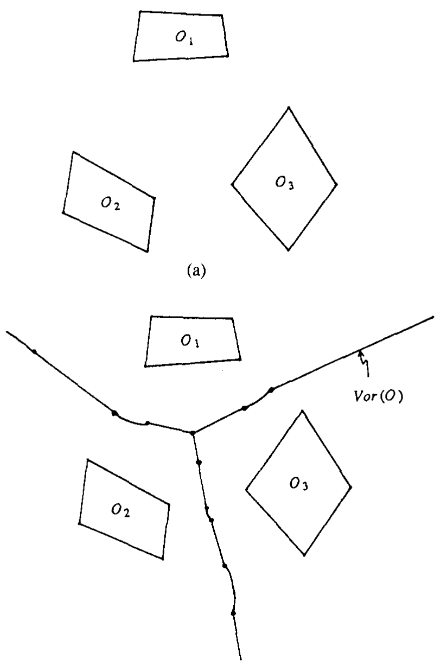

Fig. 1. The Voronoi diagram. (a) The terrain $O = \{O_1, O_2, O_3\}$. (b) Vor (O).

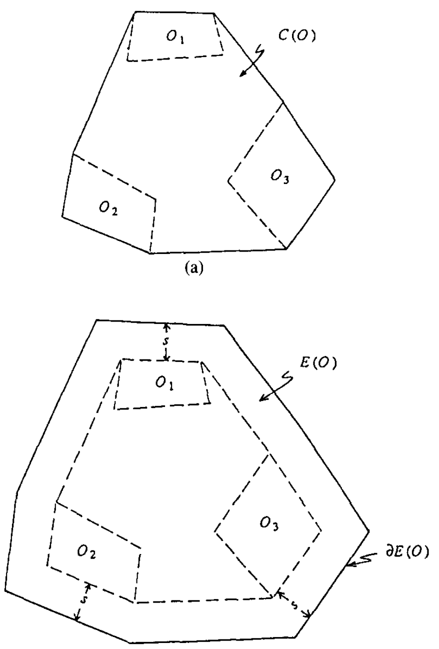

Fig. 2. Definitions of $E(O)$ and $C(O)$. The convex hull $C(O)$ of terrain O of Fig. 1(a). (b) The extended hull $E(O)$ of terrain O of Fig. 1(a).

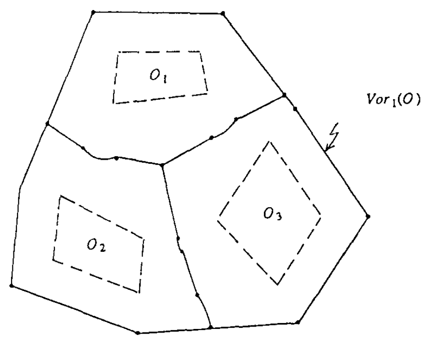

Fig. 3. The definition of Vor_1 (O) for the terrain O of Fig. 1(a).

that R navigate infinitely away from the terrain to ensure the terrain-visibility property. This problem is overcome by using Vor_1 (O), and this aspect is further elaborated subsequently.

1) Combinatorial Properties: We now estimate bounds on the number of V_1 vertices and edges. It is shown that the number of V vertices is $O(N)$ [4], [6]. Since $E(O)$ is convex, there can be at most N intersections between the edges of Vor (O) and $\partial E(O)$. Thus, the number of V_1 vertices is $O(N)$, and by Euler's equation[2] it follows that the number of V_1 edges is $O(N)$. In the present application, we need more precise bounds on the number of V_1 vertices because each V_1 vertex represents a time-consuming sensor operation.

Consider an obstacle O_i and the subset of Vor_1 (O) such that each point on this subset has a nearest neighbor on the boundary of O_i. This subset consists of one cycle and a finite set of trees (see Fig. 4). We now define the dual $D(O)$ as Vor (O) as follows: draw perpendiculars to each obstacle edge at the convex end points (obstacle vertices) and extend them outwards as shown in Fig. 5. Now Ω is partitioned into regions such that the points belonging to each partition are closer to either an obstacle or an obstacle vertex. We represent each region by a D node. Two D nodes are connected by a d edge if and only if the corresponding regions meet at either a V edge or a perpendicular. Then $D(O)$ can be shown to be a planar graph along with lines of Lee and Drysdale [7]. The dual of Vor (O) of Fig. 5 is shown in Fig. 6. We now use the dual $D(O)$ to estimate the bounds on the number of V_1 vertices and V_1 edges.

Lemma 1:

i) $(n + 5)/2 \leq \# V_1$ vertices $\leq 4N - n - 2$

ii) $3(n + 1)/2 \leq \# V_1$ edges $\leq 6N + 3n - 3$

Proof: First we prove the lower bounds. The graph of Vor_1 (O) contains $n + 1$ faces—one for each $O_i \in O$ and the other corresponding the exterior of $E(O)$. Let us count the number of edges that bound each of the faces of Vor_1 (O). By noting that each face has at least three V_1 edges, and each edge is counted no more than twice in the counting process, we have $\# V_1$ edges $\geq 3(n + 1)/2$. Now using Euler's equation for Vor_1 (O), i.e., $\# V_1$ vertices $= \# V_1$ edges $- \# V_1$ faces $+ 2$, we obtain $\# V_1$ vertices $\geq (n + 5)/2$.

To establish the upper bounds we proceed in two steps: 1) we first compute the upper bounds on the number of number vertices and edges of Vor (O) by suitably using the properties of $D(O)$; 2) we use the bounds of step 1) to estimate the bounds on the number of V_1 edges and vertices.

[2] Euler's equation for any planar graph $G = (V, E)$ with set of faces F is given by $|V| + |F| = |E| + 2$.

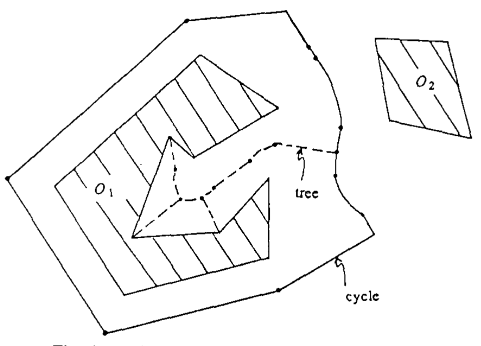

Fig. 4. Subset of Vor_1 (O) corresponding to O_1.

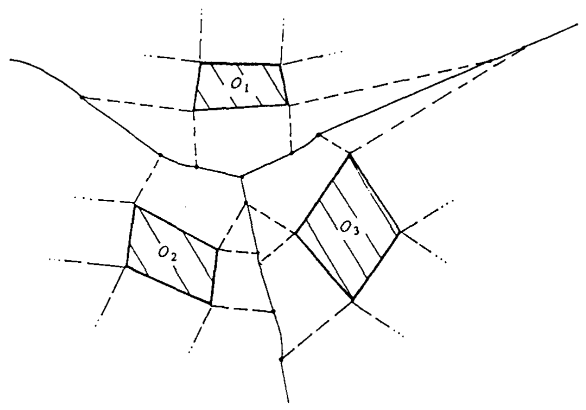

Fig. 5. Partition of Ω with V edges and perpendiculars.

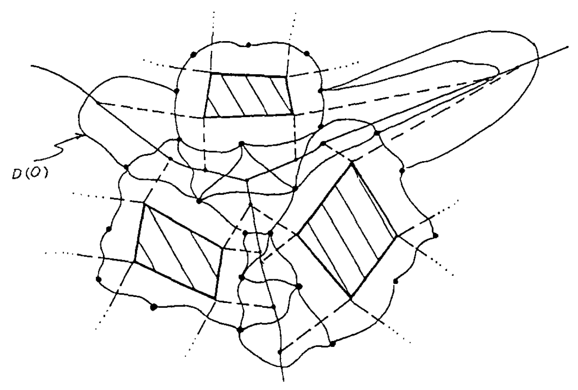

Fig. 6. Dual $D(O)$ corresponding to the partition of Fig. 5.

Consider step 1). Let V_D, E_D, and F_D represent the sets of vertices, edges, and faces, respectively, of $D(O)$. Since there can be at most two perpendiculars from each obstacle vertex, there can be at most $2N$ regions that yield d nodes. Thus, we have $|V_D| \geq 2N$.

As shown in Fig. 5, each v node is formed by the meeting of at least three segments where each segment could be either a V edge or a perpendicular. Thus, each V vertex generates a face of $D(O)$ with at least three D edges. Now for each face of $D(O)$ let us count the number of D edges bounding the face. There are three types of faces of $D(O)$: faces corresponding to V vertices, faces corresponding to obstacles, and the exterior infinite face.

a) Consider faces formed by V vertices. There are $|F_D| - n - 1$ such faces, and each face has at least three edges, since there are n faces formed by the obstacles, and there is an outer face (exterior inifinite face). Thus these faces yield a total count of at least $3(|F_D| - n - 1)$ edges.

b) Consider faces corresponding to obstacles. Each obstacle generates a face of $D(O)$. Each D edge denotes a transition from the influence of vertex to edge, or edge to vertex or edge to edge. A D edge corresponds to an edge-to-edge transition if the obstacle corner formed by the two edges is nonconvex. The D edges formed by the other two transitions correspond to convex obstacle vertices. There are N obstacle edges and at least $3n$ convex vertices. Thus, the faces generated by obstacles yield a total count of $N + 3n$ D edges in all.

c) Let the exterior face contain p edges.

In the process of counting in parts a) to c), each edge is counted at most twice. Thus, we have, $2|E_D| \leq 3(|F_D| - n - 1) + N + 3n + p = 3|F_D| + N + p - 3$. The application of Euler's equation (i.e., $|V_D| + |F_D| = |E_D| + 2$) establishes that $|F_D| \leq 3N - p - 1$ by using $|V_D| \leq 2N$.

Now, each face of $D(O)$ corresponds to precisely one of a V vertex, an obstacle or the exterior face; thus we have $|F_D| = \#\, V$ vertices $+ n + 1$. Thus, $\#\, V$ vertices $\leq 3N - n - p - 2$. To get an upper-bound on the number of V edges, we apply Euler's equation again (in the form $|E_D| \leq |V_D| + |F_D| - 2$) on $E(O)$. Hence, we have $|E_D| \leq 2N + |F_D| - 2 \leq 5N - p - 3$. Now note that $|E_D| = \#\, V$ edges $+ \#$perpendiculars (by the definition of $D(O)$), and each obstacle gives rise to at least three perpendiculars. Thus we obtain $\#\, V$ edges $\leq |E_D| - 3n = 5N - p - 3n - 3$. So far we have been able to get upper bounds on the number of V vertices and edges.

Now consider step 2). By noting that there can be at most N intersections of the V edges with the $E(O)$ and $E(O)$ has at most p edges, we obtain $\#\, V_1$ vertices $\leq 4N - n - 2$, and $\#\, \xi$ edges $\leq 6N - 3n - 3$. ■

2) Connectivity Property: A map $\text{Im}: \Omega \rightarrow \text{Vor}\ (O)$ is defined by O'Dunlaing and Yap [10] as follows: Consider $x \in \Omega$. If x is on Vor (O), then $\text{Im}(x) = x$; otherwise, $\text{Near}(x) = \{p\}$ for some point p on $\partial\Omega$, the boundary of Ω. Let L be the semifinite straight line from p through x, and define Im (x) to be the first point y (if it exists), where L intersects Vor (Ω). Intuitively, Im (x) is obtained by "pushing" x away from the closest wall (or corner) until it lies on the Voronoi diagram. We state a Theorem from [10].

Fact 1: If Ω is bounded, then i) the map Im is a continuous retraction of Ω onto Vor (O) (so Vor (O) is a retract of Ω), and ii) if $\text{Im}(x) \neq x$, then the clearance is strictly increasing along the line-segment joining x to Im (x). ■

We show the connectivity of $\text{Vor}_1(O)$ in the following lemma:

Lemma 2: The 1-skeleton Vor_1 (O) is topologically connected.

Proof: The obstacle-free region Ω is homeomorphic to (real) plane with n closed discs removed from it; each disc corresponding to a single obstacle. Hence, Ω is (polygonally) path connected. Im is shown to be a continuous retraction of a bounded Ω onto Vor (O) (Fact 1). Thus, $\text{Vor}(O) \cap E(O)$ is a continuous image of a connected set Ω (bounded appropriately), and hence is connected. Now $\partial E(O)$ is topologically connected. Now $(\text{Vor}\ (\Omega) \cap E(O)) \cap \partial E(O) \neq \phi$, and hence $\text{Vor}_1\ (O) = (\mathit{Vor}\ (\omega) \cap E(O)) \cup \partial E(O)$ is connected by the Cloverleaf Theorem (Munkres [5]). ■

3) Local Constructibility Property: In general, Vor_1 (O) does

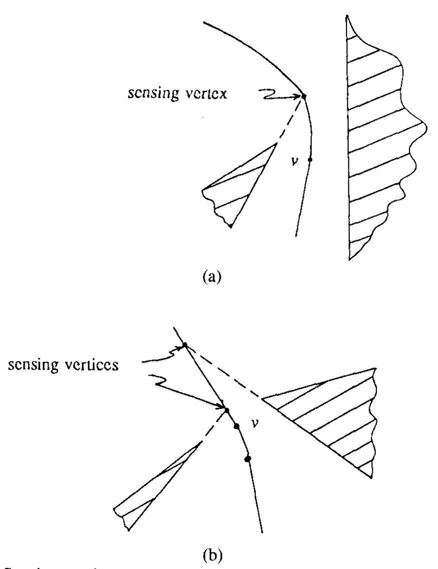

Fig. 7. Sensing vertices. (a) One sensing vertex. (b) Two sensing vertices.

not satisfy the local constructibility as shown in the example of Fig. 7(a). In this example, R has reached the V_1 vertex v by moving along the straight-line V_1 edge. After a scan operation has been performed from v, only one edge of the obstacle on the left is visible to R. With the information, the other end vertex v_1 of the parabolic V_1 edge cannot be computed, since v_1 depends on the location of the perpendicular to the hidden obstacle edge.

To alleviate the above problem, we introduce additional vertices, called sensing vertices, that are suitably computed points on V_1 edges as explained below. If R is located at a V_1 vertex v, the sensor gives the clockwise listing of the obstacle vertices and intervals of obstacle edges that are visible from v. We first compute the vertices and edge segments that are closest to v. There will be a V_1 edge emanating from v corresponding to each consecutive pair of objects found above. We can compute the equations of these edges in $O(N)$ time [4], but we may not be able to compute the end vertices of these edges as in Fig. 7. Note that this happens when at least one of the obstacle edges incident on a convex vertex is not detected by the sensor. In this case, we extend the known obstacle edge to intersect the V_1 edge at a sensing vertex. If there are two sensing vertices at any step, then we choose the one that is nearest to v along the V_1 edge. Next, R first moves to the sensing vertex and then performs a scan operation and obtains the hidden obstacle edge. By scanning from at most two sensing vertices, the other end of the V_1 edge can be computed (Fig. 7). Thus, after including the sensing vertices, Vor_1 (O) satisfies the local-constructibility property. In all, there can be at most $N-C$ sensing vertices, since each sensing vertex corresponds to one obstacle vertex and only a convex obstacle vertex can give rise to a sensing vertex.

4) Terrain Visibility Property: In order to establish the correctness of our algorithms, we require one additional property other than those discussed in Section II. This property, called terrain visibility, can be defined as follows: Every point in the closure of free space Ω is visible from some V_1 vertex, i.e., for $x \in \bar{\Omega}$, there exist a V_1 vertex v such that the line joining x and v is entirely contained in $\bar{\Omega}$.

We shall now show that Vor_1 (O) satisfies the terrain-visibility property. Let us obtain the cellular decomposition of the closure $\Omega \cap E(O)$ as follows: From each of V vertex v, draw extension lines joining v to all its nearest obstacle edges and vertices. See Fig. 8. Furthermore, join each vertex of $E(O)$ to its corresponding obstacle vertex; these lines are also called extension lines. The extension lines, V_1 edges, and obstacle edges partition the closure of $\Omega \cap E(O)$ into cells. Each cell is bounded by exactly two extension lines, exactly one V_1 edge, and at most one obstacle edge. Consider V_1 edges which are V edges. We have two basic types these V_1 edges:

Type 1: Straight-line V_1 edges: These V_1-edges are formed by two obstacle vertices as in Fig. 9(a) or formed by two obstacles edges as in Fig. 9(b). In either case, the V_1 edge, e, has two convex cells on either side, whose union is star shaped with respect to both end vertices of e. If the V_1 edge terminates on a nonconvex obstacle vertex as in Fig. 9(c), then the case is similar to one shown in Fig. 9(a).

Type 2: The parabolic V_1 edges formed by one obstacle edge e_1 and one obstacle vertex, say v (Fig. 9(d)). Consider the decomposition of the region into a triangle and a quadrilateral by joining the end vertices of e by a straight line (Fig. 9(d)). The region is convex with respect to either of the end vertices of e.

Lemma 3: Vor_1 (O) satisfies the terrain-visibility property.

Proof: Every point in the complement of $C(O)$ is visible from some vertex of $E(O)$. Consider $x \in \Omega \cap C(O)$. First, if $\text{Im}(x) \in C(O)$, then move along the corresponding V edge e along some direction until either we meet a V vertex or we move out of $C(O)$. If the former occurs, then x lies in the cell associated with e and hence is visible from its end vertices. If the latter occurs, then reverse the direction of motion along e and traverse in the other direction until a V vertex v is encountered. Second, if Im (x) does not belong to $C(O)$, then move along the corresponding V edge toward $C(O)$ until we meet $\partial E(O)$ at the intersection point y. As we traverse along Vor (O) always choose the V edge that is closest to x at V vertices (if V vertices are encountered). It is clear that the line joining x to y will be free of obstacles and hence x is visible from y. ■

Along the lines of this lemma, we can show that Vor (O) satisfies the terrain-visibility property. Consider the case where the terrain consists of two parallel finite line segments of equal length (an appropriate terrain consisting of polygons can also be designed similarly, but we stay with the present example for simplicity). The Voronoi diagram in this case corresponds to an infinite line equidistant from the obstacle line segments. Now the points that are on the "outer side" of the line segments, can be seen from the points of the Voronoi diagram that are infinitely away from the terrain. This means that the robot has to travel (on the Voronoi diagram) an infinitely long distance away from the terrain in order to see the "hinder" parts of the obstacles. This problem does not arise in Vor_1 (O) because of $E(O)$.

B. Navigation Course

For $x \in \Omega$, let Clearance (x) denote the Euclidean distance of x from a member of Near (x). Consider a subset of Vor_1 (O) given by $\{x \in \text{Vor}_1\ (O) \mid \text{Clearance}\ (x) \geq \delta\}$, and let Vor_1^* (O) be the connected component (of this subset) that contains Im (d_0). An edge of Vor_1^* (O) could be a truncated version of an edge of Vor (O), in which case we attach a vertex at the truncated end. These vertices are called the truncated vertices and the corresponding edges are called the truncated edges. Consider a concave obstacle vertex v and the V_1 edge e that is incident on v (more precisely, the end of e gets arbitrarily close to v; for practical purposes, this is treated as if e is incident on v). For $\delta > 0$, the edge e is truncated such that v is

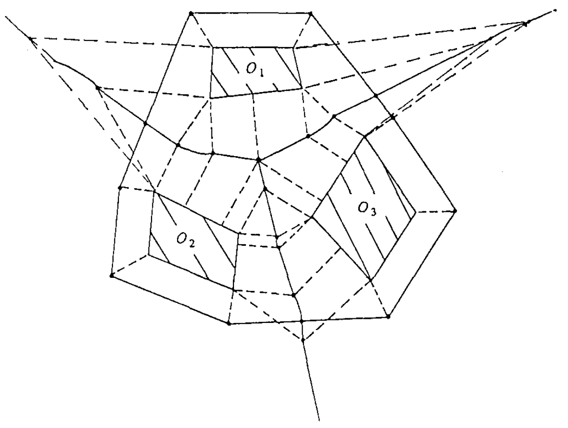

Fig. 8. Cellular decomposition of $\bar{\Omega} \cap E(O)$.

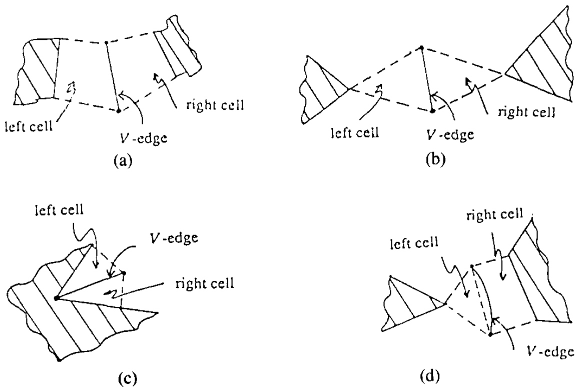

Fig. 9. Decomposition based on edges.

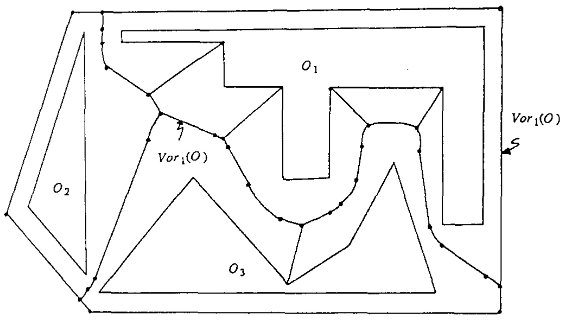

Fig. 10. The terrain $O = \{O_1, O_2, O_3\}$.

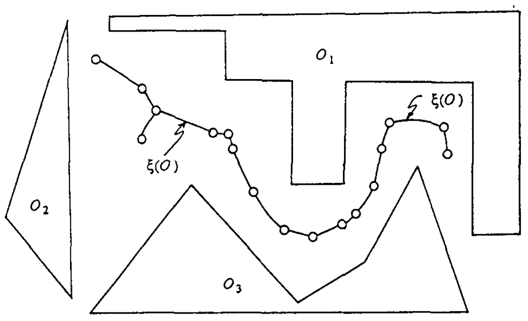

Fig. 11. $\xi(O)$ for the terrain of Fig. 10.

eliminated from $\mathrm{Vor}_1^*(O)$, since the Clearance (x) is 0 at $x = v$ and then increases as x is moved away from v along e. In this case, each the V_1 edge yields one truncated edge. We can have another case where each edge yields two truncated edges. Consider the V_1 edge e with end points v_1 and v_2 such that Clearance (x) decreases until some point and then increases as x is moved from v_1 to v_2 along e. Then, if the value of Clearance (x) falls below δ at any point on e, we will have two truncated edges (one each from v_1 and v_2). In summary, there can be at most two new truncated vertices formed out of a single V_1 edge.

Now $\xi(O)$, based on retraction, is obtained by deleting all the truncated vertices and edges from $\mathrm{Vor}_1^*(O)$ and adding the suitable sensing vertices. If $\delta > 0$, all the V_1 edges that are incident on concave obstacle vertices will be removed when $\xi(O)$ is computed. Additionally, for point robots we specifically delete all edges that are incident on concave obstacle vertices. In Fig. 10, we show the terrain $O = \{O_1, O_2, O_3\}$, and the corresponding $\mathrm{Vor}_1(O)$. In Fig. 11, we show $\xi(O)$ for a circular robot. In obtaining $\xi(O)$ from $\mathrm{Vor}_1(O)$, we add at most $N - C$ sensing vertices and delete C V_1 vertices that correspond to concave obstacle vertices. Thus, the number of ξ vertices is at most $5N - 2C - n - 2$.

Let Ψ be the maximal set of free positions of R that contains initial position d_0. Consider the set $\Gamma = \{x + y \mid x \in \Psi, y \in R\}$; here R is treated as an open disc of radius δ. The closure of Γ contains obstacle vertices, intervals of obstacle edges, and circular arcs of radius δ. The intervals of obstacle edges that are contained in $\bar{\Gamma}$ suffice to compute Ψ. Thus, the terrain-visibility property here means that every point on the obstacle boundary contained in $\bar{\Gamma}$ is visible from some V_1^* vertex. Summarizing the discussion of this section, we have the following properties.

Theorem 1: $\xi(O)$ contains at most $5N - 2C - n - 2$ vertices, and satisfies the properties of connectivity, terrain visibility, and local constructibility. ∎

IV. Navigation Algorithms

The algorithm *ACQUIRE* that solves the terrain model acquisition problem is a direct implementation of the algorithm *NAV*; the algorithm *NAV* is executed until S_2 becomes empty—at this point all ξ vertices are visited and by the terrain-visibility property the entire terrain model is available. Once the model is available, we can use the algorithm of O'Dunlaing and Yap [10] to plan a path to reach any destination point. In the view of Observation 1 and Theorem 1, we have the following theorem.

Theorem 2: The algorithm *ACQUIRE* solves the terrain model acquisition problem in a finite amount of time in at most $5N - n - 2C - 2$ scan operations with a time complexity of $O(N^2)$. After the execution of *ACQUIRE*, R can navigate to any reachable destination with a time complexity of $O(N^2)$ and with no sensor operations.

Proof: The correctness of *ACQUIRE* follows along the lines of [15], where it has been established that the four properties of finiteness, connectivity, terrain visibility, and local constructibility suffice to guarantee that the terrain model will be correctly acquired. The bound on the number of sensor operations follows from Theorem 1.

We use the adjacency list representation for $\xi(O)$. We store the coordinates of each ξ vertex in the adjacency lists. The storage

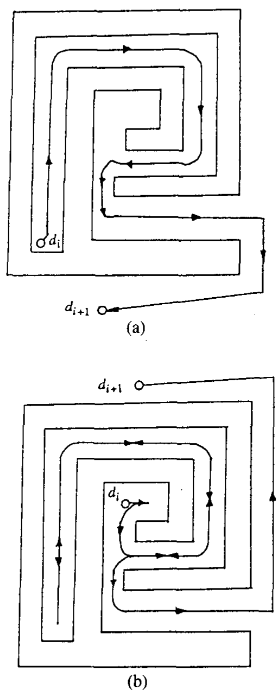

Fig. 12. Execution of *LNAV*. (a) No backtracking. (b) Backtracking.

complexity of *ACQUIRE* is $O(N)$. We use a depth-first implementation of *NAV* so that the cost of each backtracking is $O(N)$ and the cost of computation of each adjacency list of a ξ vertex is $O(N)$. Thus, the time complexity of *ACQUIRE* is $O(N^2)$. After the terrain model is built, we use the $O(N \log N)$ time algorithm of [10] to plan the subsequent navigation paths. ∎

Now consider a solution to the visit problem. The algorithm *LNAV*, that navigates R from its present location at d_i to a destination point d_{i+1} (if such path exists), is obtained by slightly modifying *ACQUIRE* as follows. After each scan, R checks to see if D_{i+1} is reachable, and moves to d_{i+1} if it is reachable. If not, the execution is continued. In the worst case, R would acquire the entire terrain model, and if d_{i+1} is not found to be reachable at this stage, then it is correctly declared as not reachable. See Fig. 12 for examples of R escaping out of a maze using *LNAV*.

LNAV completely relies on the sensor information for navigation. Since the sensor obtains only partial information about the terrain, R might navigate into local concavities as in Fig. 12(b). If R is required to navigate in regions that it has navigated in the previous traversals, it can use the previous information to plan its present course of navigation. We obtain the algorithm *GNAV* that achieves this objective as follows. We store the adjacency lists computed by R over the traversals in a global model representing a partial $\xi(O)$. The set S_2 is also stored over the traversals. Consider the navigation from d_i to d_{i+1}. *GNAV* computes a ξ vertex (on the available part of $\xi(O)$) that is reachable from d_i and moves to this vertex. Then R computes an available ξ vertex d^* that is closest to d_{i+1} according to some criterion such as distance. Then R moves along a path on $\xi(O)$ to d^*. From d^*, R uses *LNAV* to navigate to d_{i+1}. Moreover, R checks the set S_2 after every scan operation. After S_2 becomes empty, R switches off its sensor and navigates using the algorithm of [10] alone. At this stage, R has acquired a model of the terrain that is sufficient to navigate to any reachable point. At any intermediate stage, the details of the partial model of the terrain depend on the paths traversed by R in the earlier traversals. Compared to *ACQUIRE*, this solution requires at most M extra sensing operations, each operation corresponding to the position d_i, $i = 0, 1, \cdots (M-1)$. Thus, we have the following theorem (again a detailed proof follows along the lines of [15] and Theorem 2).

Theorem 3: The algorithm *GNAV* solves the visit problem. The terrain model will be completely built by R in at most $5N + M - n - 2C - 2$ scans, after which the execution of each traversal involves no scan operations and will have a time complexity of $O(N \log N)$. ∎

We can compare our method with the visibility-graph-based methods for the terrain model acquisition problem. The number of scan operations in the visibility graph based approach is at most $N - C + 1$ [16]. In the retraction method this bound is $5N - n - 2 - 2C$. The storage complexity of visibility graph based method is $O(N^2)$, whereas in the case of the retraction method, it is $O(N)$. The overall time complexity of visibility-graph-based methods is $O(N^2 \log N)$ [16], as opposed to $O(N^2)$ of the retraction method.

V. Conclusions

We presented an algorithmic framework, based on a retraction of free-space onto the Voronoi diagram, to solve two navigational problems—the visit problem and the terrain model acquisition problem—for a circular robot in unknown terrains. We presented a geometric structure, called the navigation course, based on the retraction of the terrain. The robot uses the navigation course as a road map in solving the two navigational problems. The framework presented in this paper (and also in [15]) is general in that any graph structure that satisfies the properties of finiteness, connectivity, local constructibility, and terrain visibility can be used to solve the navigation problems. It will be interesting to see if such navigation courses, based on more general retractions, can be obtained in cases such as three-dimensional terrains, polygonal robots, robots composed of linkages, etc.

Acknowledgment

We thank the anonymous reviewers for their constructive criticism that greatly improved the presentation of this paper.

References

[1] H. Abelson and A. diSessa, *Turtle Geometry*. Cambridge, MA: MIT Press, 1980, pp. 179-199.
[2] J. Cox and C. K. Yap, "On-line motion planning: Case of a planar rod," Tech. Rep. 425, Courant Institute of Mathematical Sciences, New York, Dec. 1988.
[3] S. S. Iyengar, C. C. Jorgenson, S. V. N. Rao, and C. R. Weisbin, "Robot navigation algorithms using learned spatial graphs," *Robotica*, vol. 4, pp. 93-100, Jan. 1986.
[4] D. G. Kirkpatrick, "Efficient computation of continuous skeletons," in *Proc. 29th Ann. Symp. Found. Comput. Sci.*, 1979, pp. 18-27.
[5] J. R. Munkres, *Topology: A First Course*. Englewood Cliffs, NJ: Prentice-Hall, 1975.
[6] D. T. Lee and S. Drysdale, "Generalization of Voronoi diagrams in the plane," *SIAM J. Comput.*, vol. 10, no. 1, pp. 73-87, 1981.
[7] T. Lozano-Perez and M. Wesley, "An algorithm for planning collision-free paths among polyhedral obstacles," *Commun. ACM*, vol. 22, pp. 560-570, 1979.
[8] V. J. Lumelsky and A. A. Stepanov," Path-planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape," *Algorithmica*, vol. 2, pp. 403-430, 1987.
[9] V. J. Lumelsky, S. Mukhopadhay, and K. Sun, "Dynamic path planning in sensor-based terrain acquisition," *IEEE Trans. Robotics Automat.*, vol. 6, no. 4, pp. 462-472, 1990.
[10] C. O'Dunlaing and C. K. Yap, "A retraction method for planning the motion of a disc," *J. Algorithms*, vol. 6, pp. 104-111, 1985.

[11] C. O'Dunlaing, M. Sharir, and C. Yap, "Generalized Voronoi diagrams for moving a ladder. I: Topological analysis," *Commun. Pure Appl. Math.*, vol. 39, pp. 423-483, 1986.
[12] J. B. Oommen, S. S. Iyengar, N. S. V. Rao, and R. L. Kashyap, "Robot navigation in unknown terrains using visibility graphs: Part I: The disjoint convex obstacle case," *IEEE J. Robotics Automat.*, vol. RA-3, pp. 672—681, 1987.
[13] N. S. V. Rao, S. S. Iyengar, and G. deSaussure, "The visit problem: Visibility graph-based solution" in *Proc. 1988 IEEE Int. Conf. Robotics Automat.*, pp. 1650-1655.
[14] N. S. V. Rao, S. S. Iyengar, J. B. Oommen, and R. L. Kashyap, "Terrain acquisition by a point robot amidst polyhedral obstacles," *IEEE J. Robotics Automat.*, vol. 4, no. 4, pp. 450-455, 1988.
[15] N. S. V. Rao, "An algorithmic framework for navigation in unknown terrains," *IEEE Computer*, pp. 37-43, June 1989.
[16] N. S. V. Rao and S. S. Iyengar, "Autonomous robot navigation in unknown terrains: Incidental learning and environmental exploration," *IEEE Trans. Syst. Man. Cybern.*, vol. 20, no. 6, pp. 1443-1449, 1990.
[17] J. Reif, "Complexity of mover's problems and generalizations," in *Proc 20th Ann. Symp. Found. Comput. Sci.*, 1979, pp. 421-427.
[18] J. T. Schwartz and M. Sharir, " On the piano-movers problem: I. The case of two dimensional rigid body moving amidst polygonal barriers," *Commun. Pure Appl. Math.*, vol. 36, pp. 345-398, 1983.
[19] M. Sharir, "Algorithmic motion planning in robotics," *IEEE Computer*, pp. 9-20, Mar. 1989.
[20] C. K. Yap, "Algorithmic motion planning," in *Advances in Robotics: Vol. 1: Algorithmic and Geometric Aspects of Robotics*, J. T. Schwartz and C. K. Yap, Eds. Hillsdale, NJ: Lawrence Erlbaum Associated Pub., 1987, pp. 95-144.
[21] ——, "An $O(n \log n)$ algorithm for the Voronoi diagram of a set of simple curve segments," *Discrete Computational Geometry*, vol. 2, pp. 365-393, 1987.

Performance Analysis of Total Least Squares Methods in Three-Dimensional Motion Estimation

Subhasis Chaudhuri and Shankar Chatterjee

Abstract—**An algorithm is presented to obtain the total least squares (TLS) estimates of the motion parameters of an object from range/stereo data or perspective views in a closed form. TLS estimates are suitable when data in both time frames are corrupted by noise, an appropriate model for motion analysis in practice. We analyze the robustness of different linear least squares methods for the estimation of motion parameters against the sensor noise and possible mismatches in establishing object feature point correspondence. As the errors in point correspondence increase, the performance of an ordinary least squares (LS) estimator was found to deteriorate much faster than that of the TLS estimator. The Cramér-Rao lower bound (CRLB) of the error covariance matrix is derived for the TLS model under the assumption of uncorrelated additive Gaussian noise. The CRLB for the TLS model is shown to be always higher than that for the LS model. Simulation experiments are performed to demonstrate that the proposed solution closely approximates the CRLB for the TLS model.**

Manuscript received April 13, 1990; revised May 28, 1991. This work was supported in part by the University of California at San Diego, La Jolla, CA, under University Opportunity Fund Grant 560-831-07427 and by the Naval Ocean Systems Center, San Diego, CA, under Contract SDSU N66001-87-D-0136.

S. Chaudhuri is with the Department of Electrical Engineering, Indian Institute of Technology, Bombay 400 076, Maharastra, India.

S. Chatterjee is with the Computer Laboratory for Analyzing Spatial Signals, Department of Electrical and Computer Engineering, University of California at San Diego, La Jolla, CA 92093-0407.

IEEE Log Number 9102371.

I. Introduction

The estimation of three-dimensional motion parameters plays an important role in dynamic scene analysis. It may find many practical applications such as target tracking, motion control, robot manipulation of dynamic objects, recognition of 3-D objects from a sequence of perspective views, etc. Such a broad scope of dynamic scene analysis has triggered the interest of a large number of researchers, and various algorithms have been proposed to estimate the 3-D motion parameters.

Motion estimation algorithms can be broadly classified into two categories: methods based on optical flow and those based on feature point correspondence. Although these methods have been extensively used in the literature, the performances of these methods are yet to be completely evaluated. Studies related to the sensitivity analysis for the determination of optical flow with regard to depth discontinuity and the textural variation in the image have been reported in [1].

When trying to track moving objects having a simple structure (e.g., machine parts), it is computationally advantageous to determine the motion of a few interesting object points that can be observed during the event. Using a sequence of images, the motion of a single rigid object observed under central projection has been studied in [2] and [3]. It is assumed that the exact point correspondences are known across the whole sequence. For a fixed trajectory denoting constant motion for an object, object feature point correspondence can be established [4], [5] using a path coherence constraint. We investigate quantitatively the robustness of different linear methods of motion estimation based on feature point correspondence between two frames. We consider only the least squares methods that have closed-form solutions, namely, the ordinary least squares (LS) method, the constraint least squares (CLS) method, and the total least squares (TLS) method. Some error analysis for the motion estimation algorithms based on feature point correspondence has been reported in [6] and [7]. However, they consider only the sensor noise in their data. We study the accuracy of these algorithms when there are mismatches in point correspondence in addition to having sensor noise in the data.

The motion of a rigid object may be given by an affine transform

$$\bar{q}_i = \bar{p}_i\mathbf{R} + \bar{t}$$

where $\bar{p}_i = [p_{1i} p_{2i} p_{3i}]$ is a row vector representing the coordinates of the ith point in the object, $\bar{q}_i$ is the coordinates of the same point after the motion, $\mathbf{R}$ is a 3×3 orthonormal rotation matrix, $\bar{t} = [t_x\ t_y\ t_z]$ is the translation of the object between two successive time frames. The motion understanding problem involves the estimation of the $\mathbf{R}$ matrix and the $\bar{t}$ vector.

In range imaging, 3-D coordinates of certain feature points on the object are available at each time instant. Alternatively, such 3-D data can be obtained from instantaneous stereo triangulation. The translation vector $\bar{t}$ can be easily estimated by noting the change in the centroid of the object in two successive frames, and no point correspondence need be established for this purpose. While analyzing range/stereo data, the object-centered coordinate system is assumed in this paper. When the correspondence between two data sets are given, an LS estimate of $\mathbf{R}$ can be obtained [8]-[10], but the estimated rotation matrix may not be orthonormal. A CLS method for this purpose has been proposed in [11] and [18]. These methods implicitly assume the measurement errors to be confined to a single frame only. However, due to the nature of the gathering process of the image/range data, $\bar{p}_i$ and $\bar{q}_i$ are both prone to measurement noise. In order to account for the noise in both time frames, the use