ROBOT NAVIGATION IN UNKNOWN TERRAINS OF CONVEX POLYGONAL OBSTACLES USING LEARNED VISIBILITY GRAPHS

B. John Oommen
School of Computer Science
Carleton University
Ottawa: K1S 5B6, CANADA

S.S. Iyengar and Nageswara S.V. Rao
Department of Computer Science
Lousiana State University
Baton Rouge, LA 70803, USA

R.L. Kashyap *
Department of Electrical Engineering
Purdue University
West Lafayette,IN 47907,USA

ABSTRACT

The problem of navigating an autonomous mobile robot through an unexplored terrain of obstacles is the focus of this paper. The case when the obstacles are 'known' has been extensively studied in literature. The process of robot navigation in *completely* unexplored terrains involves both learning the information about the obstacle terrain and path planning. We present an algorithm to navigate a point robot in an unexplored terrain that is arbitrarily populated with disjoint convex polygonal obstacles in the plane. The navigation process is constituted by a number of traversals; each traversal is from an arbitrary source point to an arbitrary destination point. Initially, the terrain is explored using a sensor and the paths of traversal made may be sub-optimal. The *visibility graph* that models the obstacle terrain is incrementally constructed by integrating the information about the paths traversed so far. At any stage of learning, the partially learnt terrain model is represented as a *learned visibility graph*, and it is updated after each traversal. The proposed algorithm is proven to yield a convergent solution to each path of traversal. It is also shown that the learned visibility graph converges to the visibility graph with probability one, when the source and destination points are chosen randomly. Ultimately, the availability of the complete visibility graph enables the robot to plan globally optimal paths, and also obviates the further usage of sensors.

1. INTRODUCTION

Path planning and navigation is one of the most important aspects of autonomous roving vehicles. The *find-path* problem deals with navigating a robot through a completely known terrain of obstacles. This problem is extensively studied by many researchers - Brooks [2], Lozano-perez and Wesley [6], and Oommen and Reichstein [7] are some of the most important contributors. Another interesting problem in robot navigation deals with navigating a robot through an unknown or a partially explored obstacle terrain. Unlike the find-path problem, this problem has not been subjected to a rigorous mathematical treatment, and this could be attributed, at least partially, to the inherent nature of this problem. However, this problem is also researched by many scientists - Chatila [3], Crowley [4], Iyengar et al [5], Rao et al [9], and Turchen and Wong [10], present many important results.

In this paper we discuss a technique for the navigation of a point robot in an unexplored terrain that is arbitrarily populated with disjoint convex polygonal obstacles of unknown dimensions and locations. The robot is required to undertake a number of traversals; each traversal is from a source point to a destination point. Initially no information about the obstacle terrain is available. We note that for the obstacle terrain of our problem, the availability of the visibility graph enables the planning of optimal paths from any point to any point [6]. Our approach is based on incrementally acquiring the visibility graph of the obstacle terrain. The outline of our solution is as follows: The initial traversals are based an a local navigation strategy that uses the sensor information obtained by scanning the terrain. At any stage in the navigation the terrain is characterized by a partially built visibility graph called the **Learned Visibility Graph** (LVG). The LVG is updated time-to-time by integrating the information from the sensor readings. Then we use a global navigation strategy that uses the LVG in the regions it is available, and resorts to local navigation in the regions the LVG is not available. The two key issues here are the path planning and learning. We show that the proposed technique always obtains a path, if exists, to the destination point. We also show that the LVG will converge to the VG with probability one, if the source and destination points are randomly chosen. In this paper, we present our basic results, and the details such as the correctness proof of algorithms, etc. can be found in our report [8].

Our treatment is more formal than many earlier approaches to this problem. This formal framework enables us to discuss issues such as the convergence of path planning, learning, etc., which are not very explicit in earlier approaches. Again, our problem is to be contrasted from the terrain acquisition problem [10], wherein the robot navigates with the only purpose of acquiring the terrain model. As stated earlier in our problem, the robot is required to execute a number of traversals in an unexplored terrain, and the learning phase of acquiring the LVG is a part of our solution (to make the later traversals more efficient).

The organization of this paper is as follows: Section 2 introduces the definitions and notations. The local navigation technique that incorporates learning and path planning is presented in section 3. In section 4, the power of local navigation algorithm is enhanced by incorporating backtracking. As a result the interior restriction on the obstacle terrain is relaxed. In section 5, a global navigation strategy that makes use of the existing terrain model. The important result that the learning eventually becomes complete is presented in section 6. The execution of the navigation algorithms on a sample obstacle terrain is presented in section 7.

* Research supported in part by the National Science Foundation under CDR8500022.

2. NOTATIONS AND DEFINITIONS

The robot is assumed to be a point in a plane that is arbitrarily populated with stationary convex polygonal obstacles. Initially the terrain is completely unexplored, and the robot is required to undertake a number of traversals; each traversal is from a source point to a destination point. Furthermore, the obstacles polygons are mutually non-intersecting and non-touching.

Of paramount importance to this entire problem is a graph termed as the **Visibility Graph** (VG). The VG is a pair (V,E), where

(i) V is the set of vertices of the obstacles, and.

(ii) E is the set of edges of the graph. A line joining the vertices v_i to v_j forms an edge $(v_i,v_j) \in E$ if and only if it is an edge of an obstacle or it is not intercepted by any other obstacle.

Initially the VG is totally unknown to the robot, and the robot graduates through various intermediate stages of *learning* during which the VG is incrementally constructed. These intermediate stages of learning are captured in terms of the **Learned Visibility Graph** (LVG), which is defined as follows: $LVG = (V^*,E^*)$, where, $V^* \subseteq V$ and $E^* \subseteq E$. The LVG is initially empty, and is incrementally built. Ultimately, the LVG converges to the exact VG.

We assume, throughout this paper, that the robot is equipped with a sensor capable of measuring the distance to an obstacle in any specified direction. Also, we assume that the robot is equipped with sensors which enable the navigation along the edges of an obstacle. Thus, the robot can navigate arbitrarily close to the obstacle edges. We denote the interior of any polygon ξ by $INT(\xi)$. The straight line from the point P to the point Q is denoted by $\overrightarrow{PQ}$. Further, η_{PQ} denotes the unit vector along the straight line $\overrightarrow{PQ}$.

3. LOCAL NAVIGATION AND LEARNING

When the robot navigates in a completely unexplored terrain, its path of navigation is completely decided by the sensor readings. The obstacles in the proximity of the source point are scanned and a suitable path of navigation is chosen. This localized nature of the local navigation makes a globally optimal path unattainable in a terrain with an arbitrary distribution of obstacles. However, local navigation is essential during the initial stages of the navigation.

In this section, we propose a local navigation technique that enables the robot to detect and avoid obstacles along the path from an arbitrary source point, S to an arbitrary destination point, D. The robot is equipped with a primitive motion command MOVE(S,A,λ), where (a) S is the source point, namely, the place where the robot is currently located. (b) A is the destination point which may or may not be specified. (c) λ is the direction of motion, which is always specified. If A is specified, then the robot moves from S to A in a straight line path. In this case, the direction of motion λ is the vector η_{SA}, the unit vector in the direction of $\overrightarrow{SA}$. If A is not specified, then the robot moves along the direction λ as follows: If the motion is alongside an edge of an obstacle, then the robot moves to the end point of the edge along the direction λ. This end point is returned to the calling procedure as point A as in Fig. 1(a). If motion is not alongside an edge of an obstacle, then the robot traverses along the direction λ till it reaches a point on the edge of an obstacle as shown in Fig. 1(b). This point is returned as the point A to the calling procedure.

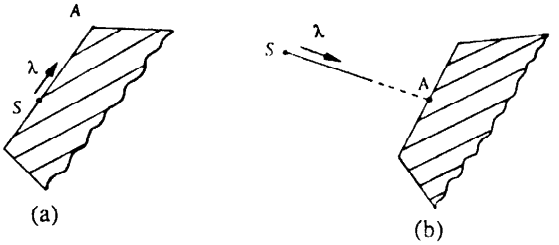

Fig. 1. Value returned by MOVE(S,A,λ), when A is not specified.

For the treatment in this section we assume that the obstacles do not touch or intersect the boundaries of the terrain R. In other words, the obstacles are properly contained in the terrain R. This is formally represented as

$$\bigcup_{i=1}^{k} INT(w_i) \subseteq INT(R) \qquad (1)$$

As a consequence of this assumption there is always a path from a source point S to a destination point D. However, this restriction is removed in the next section.

We present the procedure NAVIGATE-LOCAL that uses a *hill-climbing* technique to plan and execute a path from an arbitrary source point S to an arbitrary destination point D. The outline of this procedure is as follows: The robot moves along $\overrightarrow{SD}$ till it gets to the nearest obstacle. It then circumnavigates this obstacle using a local navigation strategy. The technique is then recursively applied to reach D from the intermediate point. Further, apart from path planning the procedure also incorporates the learning phase of acquiring the VG.

The robot moves along the direction η_{SD} till it encounters an obstacle at a point A which is on the obstacle edge joining the two vertices, say, A_1 and A_2. At this point the robot has two possible directions of motion: along $\overrightarrow{AA}_1$ or $\overrightarrow{AA}_2$ as shown in Fig. 2. We define a local optimization criterion function J as follows:

$$J = \eta_{SD} \cdot \lambda \qquad (2)$$

where λ is a unit vector along the direction of motion. Let λ_1 and λ_2 be the unit vectors along $\overrightarrow{AA}_1$ and $\overrightarrow{AA}_2$ respectively. Let $\lambda^* \in \{\lambda_1,\lambda_2\}$ maximize the function J given in equation (2). The robot then undertakes an exploratory traversal along the direction $-\lambda^*$ till it reaches the corresponding vertex called the **exploratory** vertex. At this exploratory point the terrain is explored using the procedure UPDATE-VGRAPH. Then the robot retraces along the locally optimal direction λ^* till it reaches the other vertex S^*, whence it again calls UPDATE-VGRAPH. The procedure NAVIGATE-LOCAL is recursively applied to navigate from S^* to D.

The procedure UPDATE-VGRAPH implements the learning component of the robot navigation. Whenever the robot reaches a new vertex v_i this vertex is added to the LVG.

From this vertex, the robot beams its sensor in the direction of all the existing vertices of the LVG. The edge (v_i, v) is added to the edge set E^*, corresponding to each vertex $v \in V^*$ visible from v_i. The algorithm is formally presented as follows:

```
procedure UPDATE-VGRAPH(v);
input: The vertex v which is newly encountered.
output: The updated LVG = (V*,E*).
        Initially the LVG is set to (φ,φ).
comment: DIST(v1,v2) indicates the euclidian distance
         between vertices v1 and v2, if they are visible
         to each other. This is the auxiliary information
         stored along the LVG.
begin
1.  V* = V* ∪{v};
2.  for all v1 ∈ V*−{v} do
3.     if (v1 is visible from v) then
4.        DIST(v1,v) = |v1v|;
5.        E* = E* ∪{(v1,v)};
6.     else
7.        DIST(v1,v) = ∞;
       endif
    endfor;
end;
```

The procedure NAVIGATE-LOCAL uses the motion command MOVE and the procedure UPDATE-VGRAPH during execution. This procedure is formally described follows:

```
procedure NAVIGATE-LOCAL(S,D);
Input: The source point S and the destination point D
Output: A sequence of elementary MOVE commands
begin
1.  if (D is visible from S) then
2.     MOVE(S,D,η_SD)
3.  else
4.     if (S is on an obstacle that obstructs its view) then
5.        compute {λ1,λ2}, two possible directions of motion;
6.        λ* = direction maximizing λi.η_SD;
7.        if (S is a vertex) then
8.           if ( S ∉ V* ) then UPDATE-VGRAPH(S);
9.           MOVE(S,S*,λ*);
10.       else
11.          MOVE(S,S1,−λ*); { exploratory trip to S1 }
12.          if ( S1 ∉ V* ) then UPDATE-VGRAPH(S1);
13.          MOVE(S1,S*,λ*); { retrace steps to S* }
14.          if (S* ∉ V*) then UPDATE-VGRAPH(S*);
          endif;
15.       NAVIGATE-LOCAL(S*,D);
16.    else { move to next obstacle }
17.       MOVE(S,S*,η_SD); { move to next obstacle }
18.       NAVIGATE-LOCAL(S*,D);
       endif;
    endif;
end;
```

We shall now present a Theorem from [8] that shows that the procedure NAVIGATE-LOCAL converges.

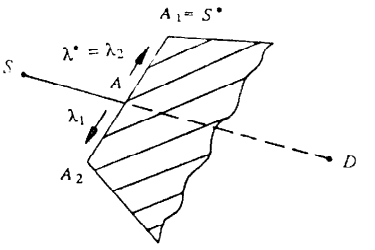

Fig. 2. The robot reached a point on the obstacle.

THEOREM 1: The procedure NAVIGATE-LOCAL always finds a path from S to D in finite time. □

Note that we have chosen to minimize the projected distance along SD by maximizing the function J in equation (2). This method may not give rise to a globally optimal path as shown in Fig. 3. Such counter examples exist for any localized navigation scheme for the want of global information about the obstacles.

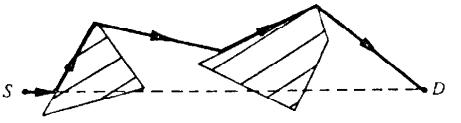

Fig. 3. Solution given by local navigation may not be globally optimal.

4. LIMITATIONS OF LOCAL NAVIGATION

The procedure NAVIGATE-LOCAL introduced in the previous section always yields a path if one exists and if the obstacles do not touch the terrain boundaries as a consequence of condition (1). The relaxation of the assumption in (1) results in two cases, in which the procedure NAVIGATE-LOCAL is not guaranteed to halt:

(a) There is no path existing between the source point S to the destination point D. Fig. 4 shows one such case. It is to be noted that in this case when the robot starts moving around the obstacle, its way is blocked in both possible directions.

(b) The angle between the obstacle edge and the terrain boundary is less than $\Pi/2$. In such a case the robot may be forced to move to the dead corner formed by the obstacle and terrain boundary (see Fig. 5).

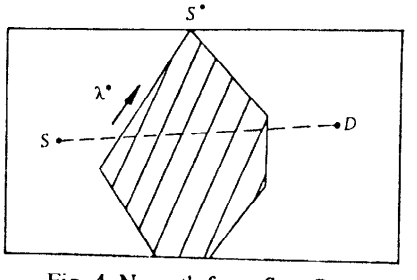

Fig. 4. No path from S to D.

In this section, we relax the condition in (1), and enhance the capability of NAVIGATE-LOCAL by imparting to it the ability to *backtrack*. The robot backtracks (by invoking procedure BACKTRACK) whenever it reaches a point from which no further moves are possible. This procedure intelligently guides the robot in the process of retracing. That is, the robot backtracks along the edges of the obstructing obstacle till

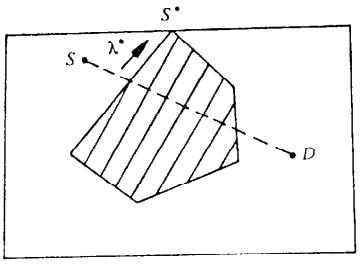

Fig. 5. Dead-corner S^*, formed by the obstacle and boundary.

an edge (S, S_1), that makes an angle less than $\Pi/2$ with η_{SD} is encountered. The fact that such an edge exists is guaranteed because of the convexity of the obstacles. The search for this edge is performed by the while loop of lines 3-6 of procedure BACKTRACK. As a result the robot moves to a point from which the NAVIGATE-LOCAL can take over. If for the same obstacle, the robot has to backtrack twice, then there is no path between S and D. In other words, if a path from S to D exists, then the robot needs to backtrack at most once along the edges of any obstacle.

---

procedure BACKTRACK(S, D, S^*);
Input: The point D is the destination point.
S is a dead corner, i.e., a vertex of an obstacle and is also on the boundary of the terrain.
Output: A sequence of MOVEs from S in such a way that if a path exis, then it can be determined using NAVIGATE-LOCAL. The location S^* is returned to the calling procedure.
begin
1. $S_1 = S$;
2. λ^* = only permitted direction of motion on the obstacle;
3. **while** $(\vec{SD}.\lambda^* < 0)$ **do**
4. MOVE(S_1, S^*, λ^*);
5. $S_1 = S^*$;
6. λ^* = only permitted direction of motion on the obstacle;
endwhile;
end;

The convergence of the procedure BACKTRACK is proved in the following theorem (see [8] for proof).

THEOREM 2: The procedure BACKTRACK leads to a solution to the navigation problem, if one exists. □

We note that if a path exists between S and D, then the robot backtracks at most once for each obstacle that lies on its way from S to D. That is because, if the procedure BACKTRACK leads the robot to another "dead-end" on the same obstacle, clearly, the robot can not navigate across the obstacle. Hence, no path exists between S and D.

Let the procedure NAVIGATE-LOCAL with the enhanced capability to backtrack be called procedure NAVIGATE--BACKTRACK. This procedure utilizes NAVIGATE-LOCAL to navigate till the robot encounters a dead-end. At this point, the procedure BACKTRACK is invoked, after which the NAVIGATE-LOCAL is resorted to. The navigation is stopped if no path exists between S and D. The formal statement and correctness proof of procedure NAVIGATE-BACKTRACK easily follow from those of NAVIGATE-LOCAL and BACKTRACK.

5. GLOBAL NAVIGATION

The procedures described in the preceding sections enable a robot to navigate in an unexplored terrain. But, the navigation paths are not necessarily globally optimal from the path planning point of view. However, the extra work carried out in the form of learning is inevitable because of the lack of information about the obstacles. Furthermore, the *LVG* is gradually built as a result of learning. In the regions where the visibility graph is available, the optimal path can be found by computing the shortest path from the source point to the destination point on the graph. The computation can be carried out in quadratic time in the number of nodes of the graph by using the Dijkstra's algorithm [1]. Such a trip can be obtained by using only computations on the *LVG* and not involving any sensor operations.

We shall now propose a technique that utilizes the available *LVG* in planning the navigation paths. In the regions where no *LVG* is available, the procedure NAVIGATE-LOCAL is used for navigation. In these regions the *LVG* is updated for future navigation. The outline of the global navigation strategy as follows:

---

procedure NAVIGATE-GLOBAL(S, D);
begin
1. Compute-Best-Vertices(S^*, D^*);
2. NAVIGATE-BACKTRACK(S, S^*);
3. Move-On-LVG(S^*, D^*);
4. NAVIGATE-BACKTRACK(D^*, D);
end

Given S and D, two nodes S^* and D^* on the existing *LVG* are computed. The robot navigates from S to S^* using local navigation. Then the navigation from S^* to D^* is along the optimal path computed using the *LVG*. Again, from D^* to D the local navigation is resorted to. Computation of S^* and D^*, corresponding to line 1 of NAVIGATE-GLOBAL, can be carried out using various criteria. We suggest three such possible criteria below:

Criterion A: S^* and D^* are the nodes of the *LVG* closest to S and D. The computation of these nodes involves $O(|V^*|)$ distance computations.

Criterion B: S^* is a vertex such that it is the closest to the line $\vec{SD}$. D^* is similarly computed. Again the complexity of this computation is $O(|V^*|)$.

Criterion C: S^* is a vertex which minimizes the angle S^*SD. Again the complexity of this computation is $O(|V^*|)$.

The closeness of the paths planned by NAVIGATE-GLOBAL to the globally optimal path depends on the degree to which the *LVG* is built. The paths tend to be globally optimal as the *LVG* converges to the *VG*.

6. COMPLETE LEARNING

Learning is an integral part of NAVIGATE-LOCAL, primarily because the robot is initially placed in a completely unexplored obstacle terrain, and the *LVG* is incrementally constructed as the robot navigates. The central goal of the learning is to eventually construct the *VG* of the entire obstacle terrain. Once the *VG* is completely constructed, the globally optimal path from S to D can be computed **before** the robot sets into motion as in [15]. Furthermore, the availability of the complete *VG* obviates the further usage of sensors. Now, we present two basic results about the learning process incorporated in our algorithm (see [8] for proof).

THEOREM 3: If no point in the free space has a zero probability measure of being a source or a destination point or a point on a path of traversal, then the *LVG* converges to the *VG* with a probability one. □

THEOREM 4: The number of sensor operations performed within the procedure UPDATE-VGRAPH to learn the complete *VG* is $O(|V|^2)$. □

In the next section we present some experimental results to illustrate the working of our technique.

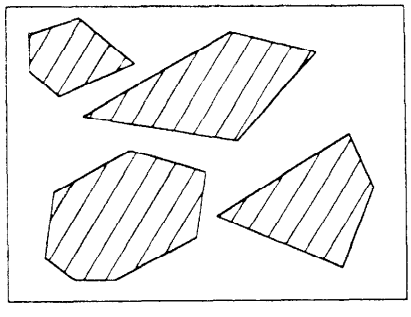

Fig. 6. Unexplored terrain.

7. SIMULATED RESULTS

In this section, we describe experimental results obtained using a rectangular obstacle terrain shown in Fig. 6. Initially the terrain is unexplored and the *LVG* is empty. A sequence of five paths is undertaken in succession by the robot. In other words, the robot first moves to 2 from 1, then to 3 from 2, etc. till it reaches 6. Fig. 7(a)-(g) illustrate the various paths traversed and the corresponding *LVG*s. Initially, during the motion from 1 to 2, the robot learnt four edges of the *VG* shown in Fig. 7(b). In the next traversal, seven more edges of the *VG* are learnt. A curve showing the number of edges learnt as a function of the number of traversals is given in Fig. 9. It is to be noted that as many as 31 out of total 39 edges of *VG* are learnt in five traversals. Suppose that at this point the global navigation strategy is invoked to navigate to 7 from 6. The S^* and D^* obtained by using criterion (A) of section 5 are shown in Fig. 8. The robot navigates locally from S to S^* then along the *LVG* from S^* to D^*, and finally locally from D^* to D. Note that the path from S^* to D^* does not involve any sensor operations, but, only quadratic time computation on the *LVG* to find the shortest path.

8. CONCLUSIONS

In this paper, we propose a technique that enables an autonomous robot to navigate in a totally unexplored terrain. The robot builds the terrain model as it navigates, and stores the processed sensor information in terms of a learned visibility graph. The proposed technique is proven to obtain a path if one exists. Furthermore, the terrain is guaranteed to become *completely learnt*, when the complete visibility graph of the entire obstacle terrain is built. After this stage the robot traverses along the optimal paths, and no longer needs the sensor equipment. The significance of this technique is the characterization of both the path planning and learning in a precise mathematical framework. The convergence of the path planning and the learning processes is proven.

REFERENCES

[1] AHO, A., J. HOPCROFT, and J. ULLMAN, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, Mass., 1974.

[2] BROOKS, R. A., Solving the Find-path Problem by Good representation of Free-space. **IEEE Trans. Systems, Man and Cybernetics**, Vol. SMC-13, No. 3, March/April 1983.

[3] CHATILA, R., Path Planning and Environment Learning in a Mobile Robot System, *Proc. European Conf. Artificial Intelligence*, Torsey, France, 1982.

[4] CROWLEY, J. L., Navigation of an Intelligent Mobile Robot, **IEEE J. Robotics and Automation**, Vol. RA-1, No. 2, March 1985, pp.31-41.

[5] IYENGAR, S.S., C. C. JORGENSEN, S. V. N. RAO, and C. R. WEISBIN, Robot Navigation Algorithms Using Learned Spatial Graphs, ORNL Technical Report TM-9782, Oak Ridge National Laboratory, Oak Ridge, August 1985, to appear in **Robotica**, Jan. 1986.

[6] LOZANO-PEREZ, T., and M. A. WESLEY, An Algorithm for Planning Collision-free Paths Among Polyhedral Obstacles, **Commun. ACM**, Vol. 22, No. 10, October 1979, pp. 560-570.

[7] OOMMEN J.B. and I. REICHSTEIN, On Translating Ellipses Amidst Elliptic Obstacles, *Proc. 1986 IEEE Int. Conf. on Robotics and Automation*, San Farancisco, California, 1986, pp. 1755-1760.

[8] OOMMEN J.B., S.S. IYENGAR, N.S.V. RAO, and R.L. KASHYAP, Robot Navigation in Unknown Terrains Using Learned Visibility Graphs. Part I: The Disjoint Convex Obstacles Case, Tech. Rep. #SCS-TR-86, School of Computer Science, Carleton University, Ottawa, Canada, Feb. 1986.

[9] RAO, N.S.V., S.S. IYENGAR, C.C. JORGENSEN, and C.R. WEISBIN, Concurrent Algorithms for Autonomous Robot Navigation in an Unexplored Terrain, *Proc. 1986 IEEE Int. Conf. on Robotics and Automation*, San Francisco, California, 1986, pp. 1137-1144. (A revised version to appear in **J. Robotic Systems**).

[10] TURCHEN M. P., and A. K. C. WONG, Low Level Learning for a Mobile Robot: Environmental Model Acquisition, to be published in *Proc. 2nd Int. Conf. Artificial Intelligence and Its Applications*, December 1985.

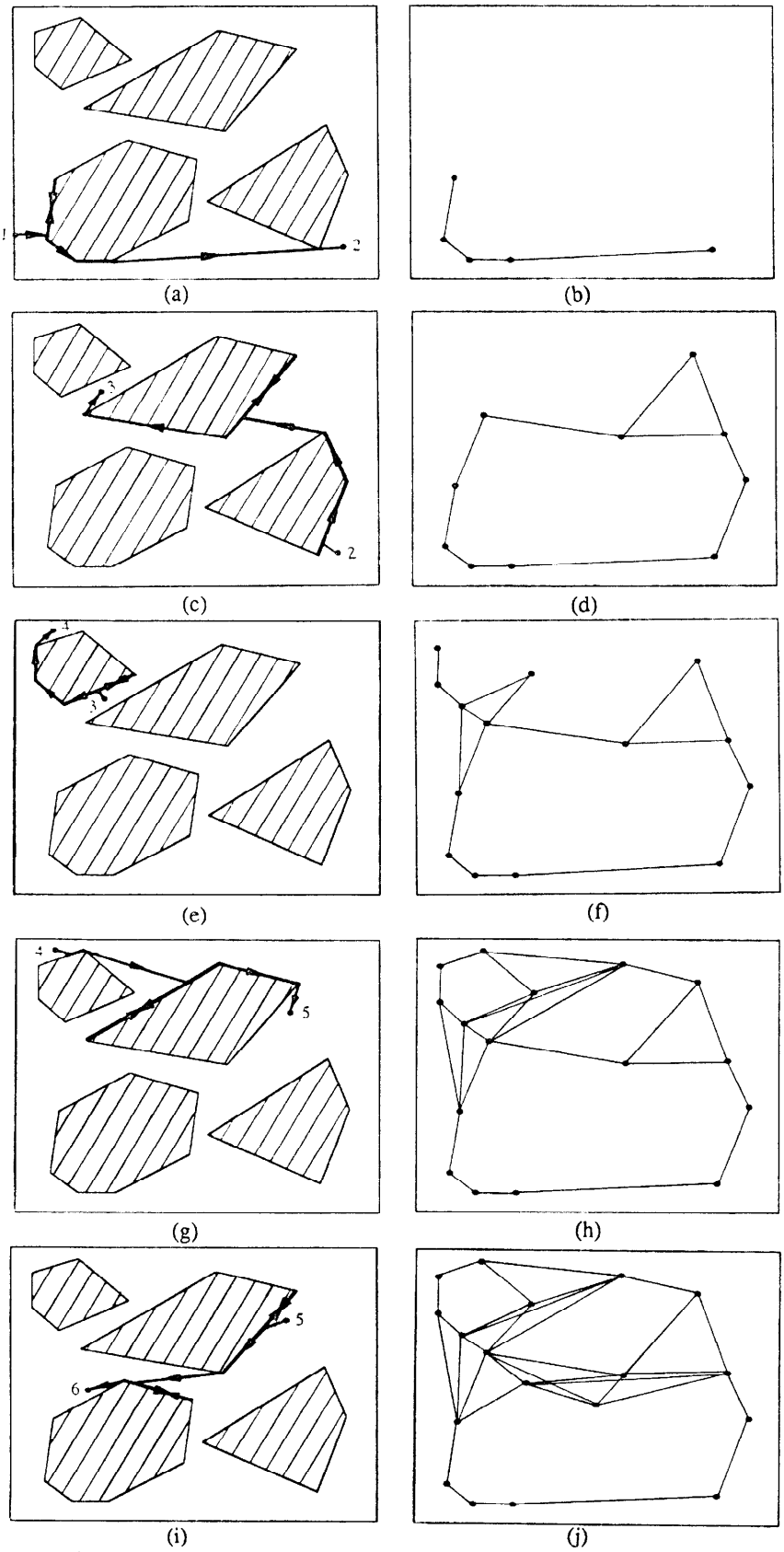

Fig. 7. Illustration of the navigation process.

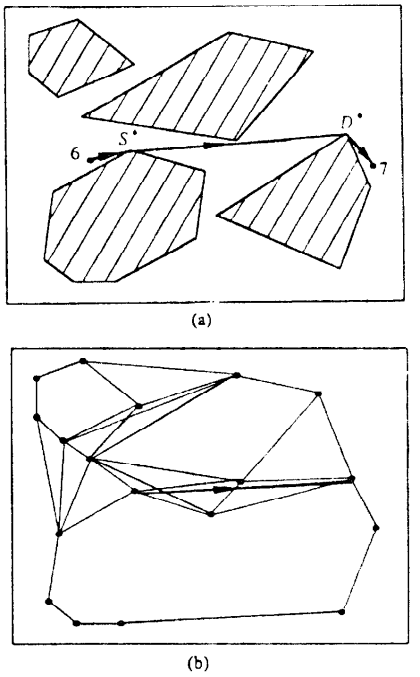

Fig. 8 . NAVIGATE-GLOBAL from 6 to 7.

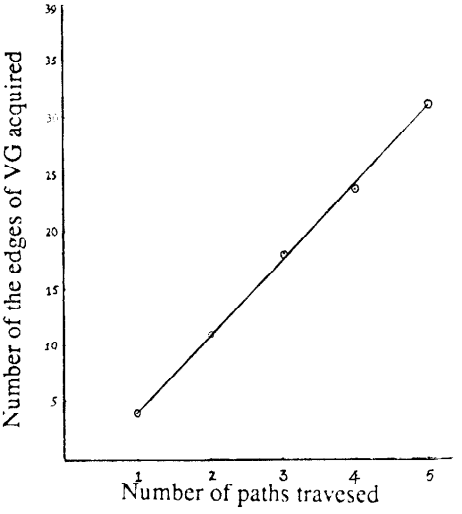

Fig. 9. Graph showing the construction of the VG