# ON TERRAIN ACQUISITION BY A FINITE-SIZED MOBILE ROBOT IN PLANE †

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

*C.C. Jorgensen* *C.R. Weisbin*
Center for Engineering Systems Advanced Research
Engineering Physics and Mathematics division
Oak Ridge National Laboratory
Oak Ridge, TN 37831

*ABSTRACT*

The *terrain acquisition problem* deals with the acquisition of the complete obstacle terrain model by a mobile robot placed in an unexplored terrain. This is a precursory problem to many well-known *find-path* and related problems which assume the availability of the complete terrain model. In this paper, we present a method for terrain acquisition by a finite-sized robot operating in plane populated by an unknown (but, finite) number of polygonal obstacles; each obstacle is arbitrarily located and has unknown (but, finite) number of vertices. The robot progressively explores newer vertices of the obstacles using sensor equipment. We show that the complete terrain model will be built by the robot in a finite time. We also show that at any point of time the partially acquired terrain suffices for the navigation of the robot during the exploration. Hence we conclude that the navigation techniques for known terrains can be applied for the robot navigation during exploration.

## 1. INTRODUCTION

*The Terrain Acquisition Problem* deals with acquiring the terrain model by an autonomously roving robot. After the terrain model is completely acquired: (A) The existing algorithms for the find-path problems can be used to plan the navigation paths. (B) Sensor equipment may not be needed for the purposes of navigation (at least in theory). Thus, the terrain acquisition could be conceived as a precursory problem to robot navigation problem in known terrains. The main motivation for this problem stems from the availability of a host of techniques for navigation in known terrains [1,3,5,6,8,12,13]. In many earlier research efforts for navigation in unexplored or partially explored terrains, the terrain acquisition is intermixed with the navigation. In these techniques, the terrain model is acquired through *incidental learning* as the robot moves in the terrain. See Iyengar et al [2], Oommen et al [9], Rao et al [9,10] for different terrain acquisition strategies.

The terrain acquisition problem discussed in this paper can be defined as follows: A finite-sized robot is placed in an obstacle terrain populated by an unknown (but, finite) number of polygonal obstacles of varied sizes and locations in the plane. We consider a robot that can be inscribed in a circle of diameter $\delta$, $\delta>0$. For instance the exact shape of the robot can be a circle, a polygon, a rod or a ladder. We consider finite obstacle terrains, i.e., there exists a circle of radius $R$ ($>0$) that contains all the obstacles. In the terrain, the envelop of width $\delta$ surrounding each obstacle is free of obstacles. Furthermore, we assume that the robot is equipped with algorithm $A$ for the navigation in a known 2 dimensional obstacle terrain of polygonal obstacles. For example, if the robot is a polygon then we may choose $A$ from the algorithms of Lozano-perez and Wesley [3], Reif [12], or Schwartz and Sharir [13], etc. If the robot is circular, then $A$ may be as described in O'Dunlaing and Yap [6]. In case the robot is a rod or a ladder $A$ may correspond to the algorithms of O'Dunlaing et al [5] or ones used for polygonal robot (as appropriate). The technique proposed in this paper is valid for the above mentioned cases. The robot is capable of translating and rotating around the center and around a point on the periphery of the circle in which it is inscribed. Moreover we assume that the robot knows its cooridinates at any point of time. The sensor is located at any point on the robot and is capable of detecting all the obstacle vertices and edges that are visible from the present location of the robot. Initially, the robot is placed at an arbitrary point in the obstacle terrain and is required to autonomously navigate and acquire the complete obstacle terrain model.

We deal with navigating the robot in the terrain with the only purpose of acquiring the terrain ( as in [10,14]). This has to be clearly contrasted from the navigation based on incidental learning. In the latter case, the navigation is goal-directed and the terrain acquisition is of secondary concern. Our solution is based on the terrain acquisition algorithm proposed by Rao et al [10] for a rather ideal version of the problem. In [10] a terrain acquisition algorithm for a point robot is given for a terrain populated by polygonal obstacles in two/three dimensions. However, this algorithm can not be directly applied to a robot of finite-size for the following reasons: (A) The sensor can not always be placed at the vertices of the obstacles as required in [10]. (B) Another important issue is the navigation of the robot using the partially built model during terrain acquisition.

In this paper, we present a method which guarantees the completion of terrain acquisition in a finite time. Our discussion here is brief and a more detailed discussion is found in [11]. This paper is organized as follows: In section 2 the terrain acquisition algorithm of Rao et al [10] is briefly discussed. This method is theoretically extended in section 3; specifically we show that the terrain acquisition will be complete in finite time. We also show that the partially acquired terrain information suffices for intermediate navigation needed during the process of terrain acquisition.

† Research sponsored by office of Basic Energy Sciences, U.S. Department of Energy under contract number DE-AC05-840R21400 with Martin Marietta Energy Systems.

## 2. BASIC TERRAIN ACQUISITION ALGORITHM

In this section we briefly present the basic terrain acquisition algorithm of Rao et al [10] for a point robot. The **Point Robot Autonomous Machine** (PRAM) of [10] is a point-sized robot equipped with a computing device and an ideal sensor capable of detecting all vertices visible from the present location of the robot. The PRAM is capable of moving in a straight line to a specified destination point from its present location. The obstacle terrain is populated by an unknown (but, finite) number of polygonal obstacles of unknown sizes and locations in plane. The terrain acquisition involves autonomously navigating the robot, and acquiring all the vertices of each obstacle (using the sensor information acquired locally time-to-time). The terrain acquisition algorithm is based on incrementally constructing the *Visibility Graph* of the terrain $O$, denoted by $VG(O)$. Formally, the $VG(O)$ is defined as a graph $(V,E)$, where (i) $V$ is a set of all vertices of all obstacles, and (ii) The line joining two vertices $u$ and $v$, $u,v \in V$, forms an edge $(u,v) \in E$ if it is not obstructed by an obstacle or it is an edge of an obstacle.

Note that the $VG(O)$ is an undirected graph and is unique for a given obstacle terrain.

The outline of the terrain acquisition algorithm of [10] is as follows: The PRAM initially starts at a point in the obstacle terrain, and scans and moves to a nearest obstacle vertex. From this starting vertex the algorithm ACQUIRE is invoked. The PRAM moves from vertex to vertex in a systematic manner (as described below in case(A) and case (B)); when a vertex is visited for the first time, a 'scan' operation is performed (a vertex may be visited again during the navigation of the robot to an unvisited vertex from its present location). Let the robot be located (for the first time) at vertex $v$. The adjacency list of $v$ (in $VG(O)$ ) is built by detecting all the vertices visible from $v$ using the scan operation. The vertex $v$ is marked 'visited' and then pushed onto a stack. We have two cases:

**Case (A):** If $v$ has unvisited adjacent nodes, then the PRAM moves to a node, say $w$, which is nearest (to $v$) among the unvisited adjacent nodes. From $w$ the algorithm ACQUIRE is recursively invoked.

**Case (B):** If all adjacent nodes of $v$ are visited, then the nodes on the stack are repeatedly popped till a node $x$ with at least one unvisited adjacent node is obtained. Then shortest paths to each of the unvisited adjacent nodes of $x$ are computed using the Dijkstra's shortest path algorithm. The PRAM chooses shortest (among the computed ones) and moves to the corresponding unvisited node $w$. From the node $w$ the algorithm ACQUIRE is recursively invoked.

The algorithm (and hence the terrain acquisition process) terminates when the PRAM is located at a vertex $u$ such that (a) all nodes adjacent to $u$ are visited, and (b) the adjacent nodes of each node on the stack (at this time) are all visited. At this point, PRAM moves back to the starting vertex along the shortest path.

**Lemma 1:**[10] The visibility graph of the terrain of polygonal obstacles in the plane is *connected*, i.e. there exists a path between any two nodes. □

**Lemma 2:**[10] The order in which the unexplored vertices of the obstacles are visited by the PRAM while executing ACQUIRE is exactly the same as the order in which the new nodes of the $VG(O)$ are visited by a depth-first-search algorithm (if $VG(O)$ were available). □

The sufficiency condition on the ACQUIRE is that a 'scanning' operation be performed from every vertex of each obstacle. The correctness of the algorithm ACQUIRE follows from Lemmas 1 and 2. The fact that a depth-first-search on a connected graph visits all the nodesis used to show that PRAM executing ACQUIRE completely builds the terrain model. There are two important conditions involved in the correctness proof of the algorithm that are of subsequent interest to our discussion: (i) the underlying graph, namely $VG(O)$, is connected, (ii) the ACQUIRE visits all nodes of $VG(O)$. We use these two points as a basis for our presentation in the next sections.

## 3. FINITE-SIZED ROBOT

The detection of all vertices visible from a given vertex is a prerequisite for the execution of the algorithm ACQUIRE. If the robot is finite-sized then it is not possible, in general, to locate the sensor at the exact location of a given vertex that is currently being visited. Consequently, the sensor is displaced from the location of the present vertex, and as a result the graph constructed based on the visibility information obtained by an ideal sensor may not be same as the $VG(O)$. Furthermore, the connectivity property - which is very vital to the completion of the terrain acquisition - may not be satisfied by the graph constructed based on the sensor readings. In this first part of this section we show that the connectivity of the constructed graph is assured if the sensor is always located on a point on the *equi-distance* line (to be defined formally) of the present vertex within a specified distance from the vertex. Then, we consider the navigation of a finite-sized robot during the intermediate stages of terrain acquisition. The point-sized robot of [10] could move along the edges of the graph, but the navigation of a finite-sized robot makes it necessary to consider the terrain information in the regions in which the robot navigates. We show, in the second half of the section, that the regions through which the robot navigates during the exploration stages are basically 'known' and hence a suitably chosen $A$ suffices for the purpose of navigation.

### 3.1. Positioning for Sensing

In the case of a finite-sized robot, the sensor may be displaced by a two dimensional vector $\vec{p}_v$ from the vertex $v$ which is currently being visited by the robot. As a result, some vertices visible from $v$ may not be visible to the sensor, and some vertices not visible from $v$ may become visible to the sensor. In Fig. 1., the sensor is located at the point $v_1$, when the vertex $v$ is visited by a finite-sized robot. As a result of the displaced sensor location we have: the vertices 4 and 5, which are visible from $v$, become invisible from $v_1$, and the vertices 6 and 7, invisible from $v$, become visible from $v_1$. Thus the graph constructed based on the sensor information may not be the same as the $VG(O)$ defined in the previous section. In fact, the constructed graph may not even be connected [11]. The algorithm ACQUIRE, in this case, is *not* guaranteed to completely

acquire the obstacle terrain model. In order that the ACQUIRE execute correctly, the graph constructed using the sensor readings has to be connected.

Let the *equi-distance line* of a vertex $v$, denoted by $EL(v)$, be a portion of the bisector line (straight line through $v$ that makes equal angles with both the edges that are incident at $v$) that extends from $v$ to the outwards of the obstacle. Let $\theta(v)$ denote the angle subtended by an obstacle at its vertex $v$. A vertex $v$ is said to form a *convex* corner if $\theta(v)<\pi$. The vertex $v$ is said to form a *concave* corner otherwise (i.e. $\theta(v)>\pi$). Let us define a function $f:V\rightarrow \bigcup_{v\in V} EL(v)$ called *sensing function*, where $V$ is the set of all vertices of the obstacles. The function assigns a unique point on $EL(v)$ for each $v\in V$, i.e. $f(v)\in EL(v)$. Now, let us define the *Modified Visibility Graph* of the obstacle terrain $O$ with respect to a sensing function $f$, denoted by $VG^*_f(O)=(V,E^*)$, as follows: (i) $V$ is the set of all vertices of the obstacles, (ii) There exists an edge $(v,w)\in E^*$ if and only if the vertex $w$ is visible from $f(v)$. In other words a sensor located at $f(v)$, which is a point on $EL(v)$, will be able to detect the vertex $w$. This visibility information is represented by the edge $(v,w)\in E^*$.

In the terrain acquisition algorithm for a point-robot the exploration of a vertex $v$ corresponds to locating the robot at $v$, and finding the vertices visible from $v$. Now, for a finite-sized robot when a vertex $v$ is visited for exploration, the robot is positioned in such a way that the sensor is located at $f(v)$ ( on $EL(v)$ ). Now, the robot constructs $VG^*_f(O)$, for some $f$, in place of $VG(O)$ constructed by the PRAM. Note that the edges of $VG^*_f(O)$ are *directed*, i.e. an edge $(v_1,v_2)\in E^*$ does not necessarily imply the existence of the edge $(v_2,v_1)\in E$. Now, note that the $VG^*_f(O)$ for a given obstacle terrain $O$, depends upon $f$ and is not unique (unlike $VG(O)$). For a given obstacle terrain $O$, there exists a *family* of modified visibility graphs, denoted by $\{VG^*_f(O)\}$ corresponding to all possible $f$s. For each $v\in V$, there are infinitely many potential images in $EL(v)$ since $f(v)\in EL(v)$, and $EL(v)$ is a subset of the real-line. Consequently the cardinality of the family of graphs $\{VG^*_f(O)\}$ may not be finite and may not even be countably infinite. We now show a result that helps us in proving the convergence of our terrain acquisition.

**Lemma 3** [11]: The $VG^*_f(O)\in\{VG^*_f(O)\}$ is connected for all $f$ such that for $v\in V$

$$||v-f(v)|| \leq \begin{cases} \delta & \text{if } \theta(v)<\pi \\ \delta/\sin(\frac{\theta(v)}{2}) & \text{if } \theta(v)>\pi \end{cases}$$

where $||v_1-v_2||$ denotes the euclidian distance between two points $v_1$ and $v_2$ in plane. □

We subsequently assume that the robot is placed at $f(v)$ satisfying the condition stated in Lemma 3 for exploring the vertex $v$. In other words, when a vertex $v$ is to be explored as per the algorithm ACQUIRE, the robot must first compute the $f(v)$ and locate the sensor accordingly at $f(v)$. If $v$ forms a convex corner, then the robot can always be positioned such that the center of the circle in which it is contained lies on $EL(v)$ of a given vertex $v$ at a distance $\delta/2$ from $v$. For ease of presentation we refer to the 'center of the circle of diameter

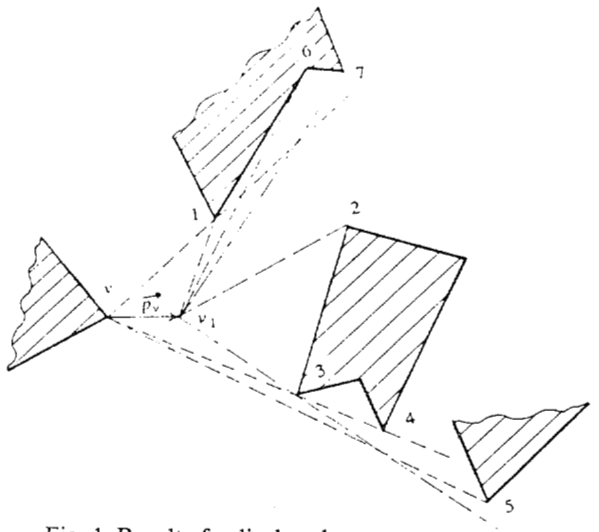



Fig. 1. Result of a displaced sensor

$\delta$ that contains the robot' as simply 'center of the robot'. At this point the robot rotates around its center so that the sensor can be located on $EL(v)$ (Fig. 2(a)). If $v$ forms a concave corner then again the robot can be similarly positioned in such a way that the sensor is no farther than $\delta/\sin(\frac{\theta(v)}{2})$ (see Fig. 2(b)). Notice that the robot located its center on $EL(v)$ so that it touches the vertex $v$ (as in Fig.2(a)) or it touches both the edges that are incident on $v$ (Fig.2(b)). Then it rotates around it center to bring the sensor onto $EL(v)$. The position of sensor achieved like this will always define $f(v)$ that satisfies the condition in Lemma 3. Any for any such $f$ the $VG^*_f(O)$ is connected. In other words $f$ specifies a sensing point for each vertex $v$ so that the constructed $VG^*_f(O)$ is connected. But, before the robot computes $EL(v)$ both the edges incident at $v$ must be known; which may not be known in general. In order to compute $EL(v)$ the robot takes an exploratory traversal around $v$ (if needed), and then positions the sensor at $f(v)$. Then the exact motion of the robot during the exploratory trip around $v$ and trip to $f(v)$ has to be computed. Both these issues are discussed in the next subsection.

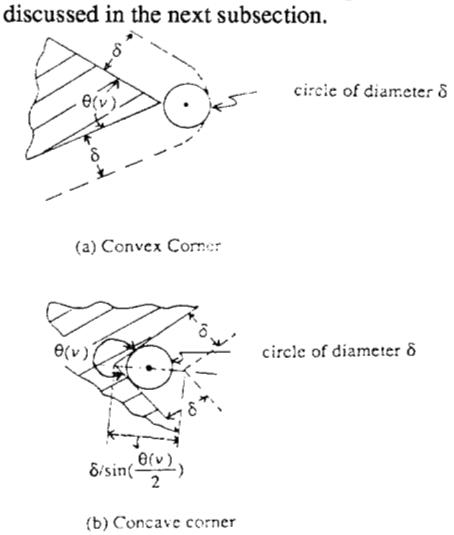



Fig. 2. Robot positioning for sensing

### 3.2. Navigation During Exploration

Now, consider the navigation of the finite-sized robot during the exploration. If the navigation is along the edges of obstacles, then the minimum clearance $\delta$ assures the collision-free navigation of the robot; as the robot can graze along the edges of the obstacles. However, if the navigation is along the other graphs edges (those edges that do not correspond to obstacles edges), then the regions around these edges have to be taken into account for the sake of navigation. We basically show that these regions are 'known', and hence algorithm $A$ can be used for navigational purposes. Another point of importance stems from the fact that the robot has to locate itself such that the sensor is placed at $f(v)$. We say that an edge incident on $v$ is *known* if a point on it other than $v$ is known. Note that if both the edges incident on $v$ are known then the line $EL(v)$ can be computed. When the robot is at an intermediate stage of exploration, it may be required to explore a vertex $v$, if one of the obstacle edges incident on $w$ is not known. At least one edge is known when $v$ is detected from some other vertex. In such a case, the robot moves into the region around $v$ to obtain the information needed to compute $EL(v)$, and it then moves to $f(v)$. The scan operation is then performed from this location. We modify the algorithm ACQUIRE as follows: Let $w$ be the vertex on the stack with unvisited adjacent nodes when the stack is accessed for finding the next univisited vertex. Note that $w$ is obtained by repeatedly popping the nodes of the stack whose adjacent nodes are all visited. Then the robot moves to $w$ and then moves to the nearest unvisited neighbor of $w$ for scan operation. The path to $w$ is planned using the Dijkstra's algorithm. Recall that the algorithm ACQUIRE computes shortest path to each of the unvisited adjacent nodes of $w$ are computed and the nearest is chosen.

Consider a vertex $v$. Exploration of $v$ involves navigating the robot from its present location to $f(v)$. Note that $f(v)$ can be any point on $EL(v)$ such that its distance from $v$ is as in Lemma 3. If one of the edges incident on $v$ are not known then the robot makes an exploratory move into the neighborhood of $v$ to explore the edges incident on $v$. It is clear that the need for the exploration of vertex $v$ occurs only when $\theta(v)<\pi$, and one of the edges incident on $v$ are not known. The robot first moves to a point such that its center lies on the perpendicular line at $v$ of the known edge as in Fig. 3(a). Navigation from the present location of the robot to this point on the perpendicular is achieved using $A$ as described subsequently in this section. After reaching this point on the perpendicular, the robot rotates around $v$ till it sees the hidden edge (Fig. 3(b)). Then it computes $EL(v)$ and moves back to $EL(v)$ by rotating around $v$ till its center of the robot lies on $EL(v)$ (Fig. 3(c)). Then the robot moves around the center till the sensor lies on $EL(v)$ as in Fig. 3(d).

Let the robot be located at $f(v)$ and is required to navigate to $f(u)$, where $u$ is another explored vertex, i.e. both the vertices incident at $u$ are known. The navigation is along the edges of the partially built $VG^*_f(O)$ is carried out as follows: If the edge $(v_1,v_2)$ of partially-built $VG^*_f(O)$ is an edge of an obstacle, then the robot navigates along the edge; the clearance $\delta$ accounts for the finite-size of the robot. If the edge $(v_1,v_2)$ is not an obstacle edge then consider the rectangle of width $2\delta$ and with $f(v_1)$ and $v_2$ as midpoints of the opposite sides as

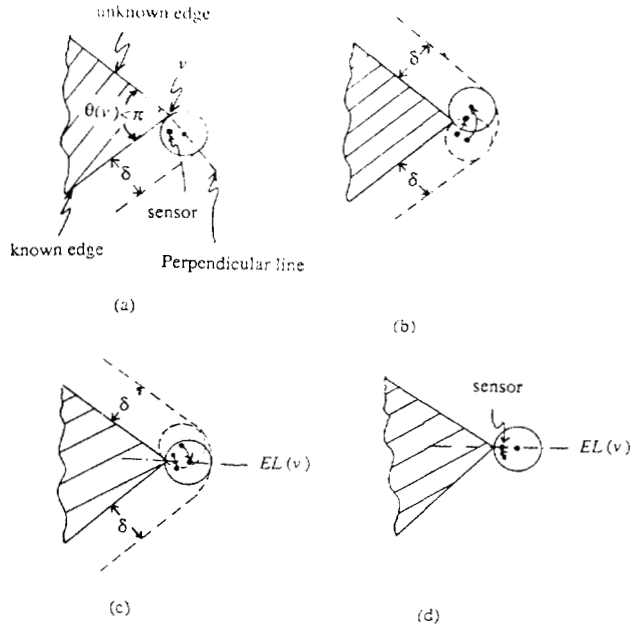



Fig. 3. Exploration of vertex $v$

shown in Fig.4. Consider the strip formed out of this rectangle by replacing the side containing $v_2$ by arc $cd$ which is the boundary of the obstacle-free envelop of the obstacle that contains $v_2$. as shown in Fig.5. Let this strip enclosure be called

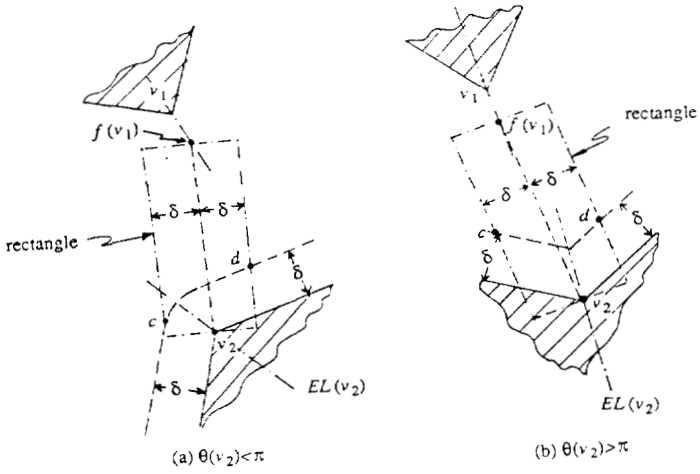



Fig. 4. The rectangle

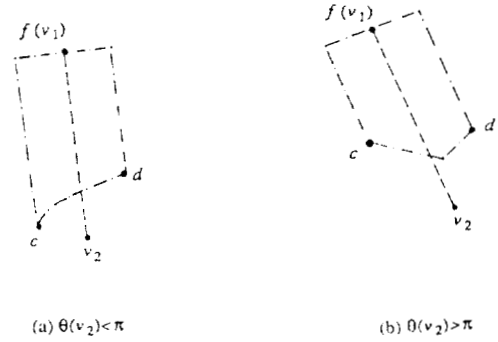
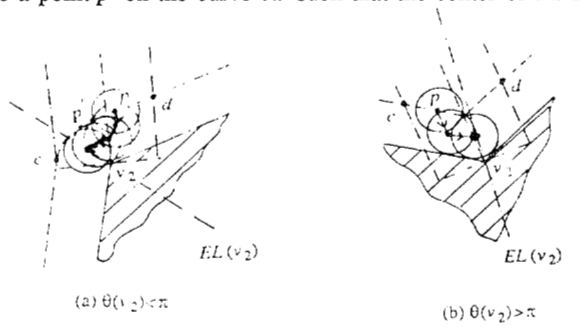



Fig. 5. The capsule $CP(v_1,v_2)$

*capsule*, denoted by $CP(v_1,v_2)$. The straight line joining $f(v_1)$ to $v_2$ is free of obstacles. So any obstacle lying in the capsule will push the robot at most to the other side of this line. But the circle that surrounds the robot still stays within the capsule. As a result there exists a path for the robot to move from $f(v_1)$ to a point $p$ on the curve $cd$ such that the center of the robot




(a) $\theta(v_2)<\pi$ (b) $\theta(v_2)>\pi$

Fig. 6. Robot motion from a point on the curve $cd$ to $f(v)$

lies on $cd$ (Fig. 6). The point $p$ exists as a result of the clearance $\delta$ and the fact that no obstacle lies on the line joining $f(v_1)$ and $v_2$. If $\theta(v_2)>\pi$ then, from this point $p$ the robot moves parallel to the line joining $f(v_1)$ to $v_2$. Then it slides along the edges of the obstacle till the circle touches $v_2$. Then it rotates around $v_2$ as in Fig. 6(b) to position itself on the required point on $EL(v_2)$. If $\theta(v_2)<\pi$ then let $p'$ be the intersection point of $cd$ and the periphery of the circle that contains the robot and is nearest to the line joining $f(v_1)$ and $v_2$. Then the robot rotates around $p'$ into the obstacle free region till its center coincides with the line joining $f(v_1)$ and $v_2$. Then it rotates around $v_2$ till it aligns the center along $EL(v_2)$ and then rotates around its center to place the sensor on $EL(v_2)$. This process is shown in Fig.6(a). Note that this navigation from $p$ to $f(v_2)$ can be carried out by utilizing the information about $v_2$ and the edges incident on $v_2$. For navigation from $f(v_1)$ to the required point $p$ on $cd$ it is enough to consider the vertices that lie inside $CP(v_1,v_2)$. If $CP(v_1,v_2)$ contains no vertices then the robot moves directly to $f(v_2)$. If $\theta(v_2)>\pi$, any vertex $w$ seen from $v$ and contained in $CP(v_1,v_2)$ will be closer to $v_1$ than $v_2$, and would have been explored earlier to $v_2$. If a vertex $x$ is in $CP(v_1,v_2)\cup CP(w,v_2)$ but hidden by a an obstacle with a vertex $w$, is also explored earlier to $v_2$ as it is nearer to $w$. On the other hand if $\theta(v_2)<\pi$, then consider the circular arc $ef$ of radius $||f(v_1)-v_2||$ with center at $f(v_1)$ superimposed on the rectangle as shown in Fig.7. This radius is at least $\delta$.

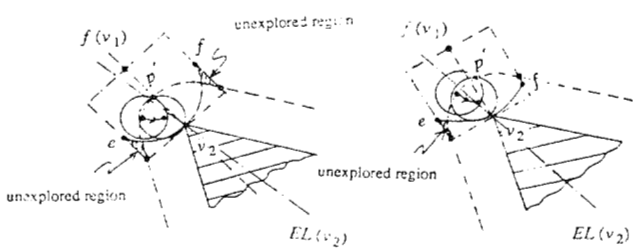



Fig. 7. The unexplored regions are not entered by the robot during its navigation to a vertex to be explored

The obstacle vertices that lie in the shaded regions may not be visited by the robot as they are farther than $v_2$ from $f(v_1)$. The radius described by any point on the periphery of the robot describes a circular arc around $p'$ whose radius is at most $\delta$. Thus the region 'swept' by the robot during its rotation around $p'$ is contained in the rectangle with the $v_2$ end replaced by $ef$. After this rotation the robot is in obstacle-free region. Thus, as shown in Fig.7 the robot will not enter the shaded regions ( possibly unknown regions ) during navigation. Thus the algorithm $A$ can now be used to plan a path from $f(v_1)$ to required point $p$ on $cd$. From this point on $cd$ the navigation is carried out as explained above. In summary, we described a method to navigate the robot along the path specified by a sequence of edges of $VG^*_f(O)$ consisting of all visited vertices.

Now, consider navigation to an unexplored vertex $v_2$ from an explored vertex $v_1$. As described in section 2, the vertex $v_2$ could be nearest among the unvisited neighbors of $v_1$ or obtained as a result of popping the nodes on the path stack till a vertex $w$ with unvisited neighbor vertices is found. In this latter case $v_2$ is nearest unvisited vertex of $w$. In this case the robot moves to $w$ from $v_1$ along a path specified by the edges of the $VG^*_f(O)$ (obtained by the Dijkstra's algorithm). At this point the robot navigates from the visited vertex $w$ to an unvisited vertex $v_2$. Hence, it suffices to consider the case where the robot moves from a visited vertex $v_1$ to an unvisited vertex $v_2$. If both the edges incident on $v_2$ are known then navigation is same as explained in the preceding paragraph. If not, say the robot navigates to a point $v^*_2$ which lies on the perpendicular to the known edge at $v_2$. Now again consider the capsule $CP(v_1,v_2)$ again. We can show that the robot navigates to $v^*_2$ by utilizing $A$ and rotational motions as described in the earlier paragraph. Thus the algorithm $A$ can be applied here also.

Now let us extend the algorithm ACQUIRE, with the capability to (a) navigate a finite-sized robot (as described in the preceding paragraphs), and (b) visit $f(v)$ whenever $v$ is to be visited. Let the new algorithm be called ACQUIRE1. Also note that ACQUIRE1 builds $VG^*_f(O)$ in the place of $VG(O)$ built by ACQUIRE such $f$ satisfies the condition stated in the Lemma 3. Now we have the following theorem.

**Theorem 1**: The algorithm ACQUIRE1 builds the complete obstacle terrain model in finite time.

**Proof**: The algorithm ACQUIRE1 visits the unexplored vertices in exactly the same order as ACQUIRE, i.e. in the order of depth-first-search. Thus the Lemma 2.2. holds for the algorithm ACQUIRE1. By Lemma 3.1 the $VG^*_f(O)$ is connected. Thus the Theorem is true from the fact the depth-first-search on a connected graph visits all nodes. □

## 4. CONCLUSIONS

We have presented a method for terrain acquisition by a finite-sized robot placed in a terrain populated by polygonal obstacles in the plane. We have theoretically established the correctness of the method. A restricted version of this terrain acquisition algorithm is implemented on HERMIES-II robot at Oak Ridge National Laboratory. Handling of errors and imprecissions associated with the real-life robots and sensors has to be given serious consideration. This is a very important aspect

of robot navigation and general methods to handle this issue are extremely useful for practical implementation of many theoretical algorithms.

**Acknowledgements**

We acknowledge the support of O.Manley of the Office of Basic Energy Sciences and the encouragement of Alex Zucker and F.C. Maienschein. We thank the anonymous referees for their valuable comments which greatly improved the presentation of the paper.

**REFERENCES**

[1] 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.

[2] IYENGAR S.S., C. C. JORGENSEN, S. V. N. RAO, and C. R. WEISBIN, Robot navigation algorithms using learned spatial graphs, **Robotica**, Vol. 4, Jan. 1986, pp 93-100.

[3] 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.

[4] LUMELSKY, V.J., and A.A. STEPHANOV, Effect of uncertainity on continuous path planning for an autonomous vehicle, *Proc. 23rd Conf. on Decision and Control*, Las Vegas, Nevada, December 1984.

[5] O'DUNLAING, C., M. SHARIR, and C. YAP, Generalized voronoi diagrams for moving a ladder I: Topological considerations, Technical Report, Courant Institute of Mathematical Sciences, New York, 1983.

[6] O'DUNLAING, C., and C. YAP, A 'Retraction' method for planning the motion of a disc, **Journal of Algorithms, Vol.6, 1985, pp.104-111.**

[7] OOMMEN, J.B., S.S. IYENGAR, S.V.N. RAO, and R.L. KASHYAP, Robot navigation in unknown terrains of convex polygonal obstacles using learned visibility graphs. *Proc. Nat. Conf. on Artificial Intelligence AAAI-86*, pp. 1101-1106, to appear **IEEE J. of Robotics and Automation.**

[8] PAPADIMITRIOU, C.H., An algorithm for shortest-path motion in three dimensions, **Information Processing Letters**, Vol. 20, 1985, pp. 259-263.

[9] RAO, S.V.N., S.S. IYENGAR, C.C. JORGENSEN, and C.R. WEISBIN, Robot navigation in an unexplored terrain, **J. Robotic Systems**, Vol.3, December 1986.

[10] RAO, N.S.V., S.S. IYENGAR, J.B. OOMMEN, and R.L. KASHYAP, Terrain acquisition by a point robot amidst polyhedral obstacles, To appear in *Proc. 3rd IEEE Conf. on AI Applications*, Orlando, Fl, Feb. 1986.

[11] RAO, S.V.N., S.S. IYENGAR, C.C. JORGENSEN, and C.R. WEISBIN, The terrain acquisition by mobile robots: II. Finite-sized robot in plane, Tech. Rep. #86-026, Dept. of Comp. Sci., Louisiana State Uni.

[12] REIF, J., Complexity of the mover's problem and generalizations, *Proc. 20th Symp. on Foundation of Computer Science*, 1979, pp. 421-427.

[13] SCHWARTZ, J.T., and M. SHARIR, On the piano movers' problem I: The special case of a rigid polygonal body moving amidst polygonal barriers, **Communications Pure Applied Mathematics**, Vol. 36, 1983, pp. 345-398.

[14] TURCHEN M. P., and A. K. C. WONG, Low level learning for a mobile robot: Environmental model acquisition, *Proc. 2nd Int. Conf. Artificial Intelligence and Its Applications*, December 1985.