Robot Navigation through Obstacles of General Shapes Using a Center-Line Oriented Algorithm

Chun-Hung Lin and *Li-Chen Fu*

Department of Computer Science & Information Engineering
National Taiwan University, Taipei, Taiwan, R.O.C.

ABSTRACT

In this paper, the problem of navigating a mobile robot around barriers in an unexplored terrain is studied. All the obstacles within the terrain are not limited to be of polygonal shapes nor to be convex. A model map is used to memorize the configuration of the environment observed so far and is updated while the robot is being navigated. With "safety" as a more important factor to the solution of the problem, an algorithm which tends to find a "center-line path" among obstacles is proposed. The case where the sensor has only limited effective range is also considered. Detailed proof is provided to assure the collision-free and goal-convergent properties of the algorithm.

1. Introduction

Path planning and *navigation* are two of the most vital issues in robotics regarding autonomous mobile robots. The former can be thought of as moving a mobile robot through a terrain populated with obstacles which are known *a priori*. There have been considerable research work on this aspect. The earliest one may be traced back to [17], and after that came in the work [2-3][8][12-13][22] [24][26]. With the obstacle model available ahead of the real maneuvering of the mobile robot, finding an *optimal* path is usually attempted and may be quite achievable. The latter is, however, a problem where some kind of "learning" of its environment has to be performed due to the lack of prior knowledge of the obstacle model while navigating the mobile robot through a terrain as before. So the problem can be thought of as the combination of *terrain acquisition* [21] and path planning [20]. The task of learning can be accomplished via a use of sensors, such as, visual sensors, laser range-finders, proximity sensors, etc. Since the circumstance here is generally much more adverse than that in the former case, finding just one path leading the robot from the source to the goal through obstacles is usually pursued. Several results have been reported in this regard, for example, [1][4-7][10-11][14-20][23][25][27-28].

The algorithm pursued in this paper is mainly motivated by the work of [19] and is a generalization of [11]. But, the approach here is different from [19] in that a *center-line* oriented path is generated whereas the path found there consists of *vertex-to-vertex* segments and portions of obstacle edges. The former is generally considered to be safer ([22]) than the latter regarding to the risk of possible collision due to the uncertainty in the sensor devices and positioning mechanisms of the robot. Also, more general obstacles can be handled in the former than in the latter which solves typically the case with polygonal obstacles. Moreover, considerable time may be spent in traversing along edges of obstacles since a "guarded" motion is usually considered slow in real application. The algorithm to be proposed here focuses on the planning level rather than the low level sensing issues, which heavily depends on the modern technology.

This paper is organized as follows: in section 2, a navigation problem in a two dimensional domain is formulated; in section 3, a center-line oriented navigation algorithm is presented along with a rigorous proof assuring its obstacle-avoidance and goal-convergence; in section 4, to illustrate the performance of the algorithm, some computer simulation examples are provided; in section 5, some modification is given on the algorithm in view of the limited distance measurability of the sensor device and nonzero volume of the robot; in section 6, generalization of the case of polygonal obstacles to the case where obstacles can be of arbitrarily shapes is made; a discussion on the difference between the current work with that of [21] is given in section 7; finally, a conclusion is reached in section 8.

2. Problem Statement

The generic nature of the problem for navigating a mobile robot around barriers is the lack of prior model of the terrain which is to be explored. Some sort of recovering of the model has to be done while the robot is heading toward the goal. This update of the model map can be accomplished through the use of several types of sensors [6][9][14][18][25][27]. Imperfect sensor readings due to limited distance measurability may, however, exist, but for the time being we will first relax this practical limitation and reconsider the nonideal case later. In the algorithm we assume the robot volumeless, i.e. the robot is assumed to be only a point. However, the similar technique can be applied to any nonzero volume robot, which will also be discussed later.

Consider the terrain with polygonal obstacles as shown in Fig. 2.1. The objective of the navigation task is obviously to lead the (point) robot from the source S to the goal G without colliding with any obstacles. Note that when time comes to update the model of obstacles from the current position of the mobile robot, a suitable sensor scans the environment and its readings enable the robot to determine the locations of visible vertices and edges of obstacles. It is clear that any two such vertices which are connected by such an edge belong to the same obstacle. The path, depending crucially on these sensor readings and on the ever-known goal direction, is then determined by some navigation algorithm.

In the situation where some obstacles may not be polygonal as shown in Fig. 6.1, the so called "tangent points" instead of vertices are detected by the sensor. Likewise, every two such points which are connected by a continuous curved edge belong to the same obstacle. The formulation of the path-searching problem is then similar to the one just described. After the robot has reached several observation points so that sufficiently many tangent points are collected, a quite close approximation of the non-polygonal obstacles by polygonal substitutes is obtained. Consequently, the solution to this problem should also be similar to the one which resolves the above problem.

3. Center-Line Oriented Navigation Algorithm

In this section, we present an algorithm which navigates the mobile robot from the source to the goal without any collision with obstacles. As indicated in section 2, positions of visible vertices of polygonal obstacles are obtained from sensor readings. This information, then, plays a crucial role in our algorithm.

The algorithm is different from that in [13] or [19] where "edge following" about obstacles is a necessary step in their algorithms. That particular procedure has an effect of shortening the geometric distance of the output path [17]. In light of a possible collision with obstacles due to some uncertainty of sensor devices or positioning mechanisms of robots, here, however, we consider "safety" as a more important factor than "shortness". In other words, the mobile robot will be kept away from any obstacle at a much safer distance while the robot is moving toward the goal point. It turns out to be a more popular criterion in the real world. Hence, our algorithm is aimed at providing a safer path which tends to follow the *center line* among obstacles. With this type of path, another possible payoff could come from the fact the mobile robot may run at a faster speed if the computer processing time of the path search is much less than the actual running time when the robot is performing an "edge following"! The following are some necessary assumptions.

Assumptions:

(A1) The robot is able to determine the direction of the goal from any position of the environment.

(A2) All polygonal obstacles are of finite dimensions. They can be *convex* and *non-convex*.

(A3) The robot is able to detect its current position in an appropriate coordinate frame.

Definition 3.1: (Track-Tree)

Track-Tree is a data structure which is used to record in every step the position of the mobile robot, or so called "intermediate goal point". In this structure, no loop is allowed, and all ascendantship as well as descendantship are clearly registered.

Definition 3.2: (Candidate-Queue)

Candidate-Queue is a priority queue which is used to record the next candidate positions at the current "intermediate goal point". Every intermediate goal position has a Candidate-Queue associated with it.

Remark: If the Candidate-Queue of some intermediate goal is empty, it means that there is no suitable next candidate point for the robot to go to. We call the intermediate goal a "*dead node*". At this point, only backtracking is allowed.

Definition 3.3: (Global-Map)

Global-Map is a disconnected graph which is used to record the global environment that the robot has observed so far. Specifically, the data in Global-Map are locations of the vertices observed so far and the relation of adjacency of these vertices.

Remark: Global-Map records locations of the vertices and the edges which have been observed by the robot. It is gradually built by integrating information about obstacles. When information about the environment structure is sufficiently collected, a better path can then be generated [21].

Definition 3.4: (Cluster of Visible Vertices (CVV))

From the sensor readings, all the visible vertices are grouped into clusters where any two vertices of a cluster are connected by an edge or edges pieced together. Edges along with vertices of each cluster form a connected subgraph and all these subgraphs form a disconnected graph.

Notation:

(1) $ang(p,q)$ denotes the angle between the two vectors $\vec{rp}$ and $\vec{rq}$, where r represents the current position of the robot.

(2) $dist(p,q)$ denotes the distance between position p and q.

(3) $V_i(p^i,q^i)$ is used to denote the ith cluster of vertices with p^i and q^i as the outwardmost vertices (i.e. both degree of vertices p^i and of q^i are one) and p^i and q^i are chosen so that q^i and p^{i+1} are close-by. Note that $ang(p^i,q^i)$ may be greater than π.

Remark: If the ith cluster contains only one vertex, then p^i and q^i are the same point. Referring to Fig. 2.1, we obtain three clusters of vertices, namely, $V_1(p^1,q^1)$, $V_2(p^2,q^2)$ (here, $p^2=q^2$), and $V_3(p^3,q^3)$. In Fig. 3.1, because the robot sits on the vertex, the clusters is $V_1(p^1,q^1)$, $V_2(S,S)$, and $V_3(p^2,q^2)$.

Navigation Algorithm:

```
Step1:If the goal is visible,
    then go to the goal directly and exit the algorithm.
Step2:Scan the environment and add new information onto
    Global-Map.
Step3:If no obstacle can be observed because the robot either
    already sits on a vertex or on an edge of an obstacle,
    then move the robot along the edge in a direction which is
    closer to that of $\vec{rG}$ till it reaches a vertex, add the location
    data of the vertex into Track-Tree, and go to Step 1,
    else perform the procedure Find_Intermediate_Goal.
Step4:If the next intermediate goal is found,
    then move the robot to that point directly, generate a node
    in Track-Tree, remove the node corresponding to the current
    position from all Candidate-Queues's constructed so far and
    go to Step 1,
    else perform the procedure Backtrack_Track-Tree.
Step5:Go to Step 1.
```

Remark : The backtracking procedure will not be necessary when the environment is free of obstacles with non-convex shapes. In fact, the procedure will be invoked only when the robot is or going to be trapped inside the concave region embraced by a non-convex obstacle.

The navigation algorithm shown above assumes that the effective range of the sensor device is unlimited. This assumption will be relaxed in section 5 and the modified version of the algorithm will also be shown. There are two procedures which are used in the algorithm and will be explained in the following: one is Find_Intermediate_Goal, and the other is Backtrack_Track-Tree.

Procedure 1: (Find_Intermediate_Goal)

Let r denote the current location of the robot.

```
Step1:Find cluster of visible vertices, $V_i(p^i,q^i)$, i = 1, 2,..., n.
Step2:Create a Candidate-Queue.
Step3:When $n=1$,
    if both $p^1$ and $q^1$ already appeared in Track-Tree,
    then go to Step 5,
    else return that the intermediate goal is $p^1$ and go to Step 6
    if (i) $q^1$ already appeared in Track-Tree; or (ii) $ang(p^1,G) <$
    $ang(q^1,G)$; or (iii) $ang(p^1,G) = ang(q^1,G)$ and $dist(r,p^1)$
    $> dist(r,q^1)$, and vice versa.
    If it turns out that the intermediate goal is $p^1$ and $q^1$ has not
    appeared in Track-Tree,
    then add $q^1$ to the Candidate-Queue associated with current
    location r, and vice versa.
```

```
Step4:Let $c_0=p^1$, $c_n=q^n$, and $c_i=\frac{(q^i+p^{i+1})}{2}$ ( i.e. $c_i$ is the middle
      point between $q^i$ and $p^{i+1}$ ) for i = 1, 2,...., n-1.
      If all $c_i$, i = 0, 1,..., n, have already appeared in Track-Tree,
      then go to Step 5,
      else if $c_i$, i = 1, 2,..., n-1, have already appeared in Track-Tree,
           then return that the intermediate goal is $c_0$ or $c_n$
                depending on which of the two $ang(c_0,G)$ and
                $ang(c_n,G)$ is smaller (if equal, consider which of the
                two distances, $dist(c_0,r)$ and $dist(c_n,r)$, is larger),
           else return that the intermediate goal is $c_j$ if $c_j$ has not
                appeared in the Track-Tree and $ang(c_j,G)$ is the smallest
                angle among $ang(c_i,G)$ where $c_i$ has not appeared
                in the Track-Tree, $1 \le i \le n-1$ (if the smallest one is not
                the only one, consider the distance from $c_j$ to r).
      Add $c_i$, $1 \le i \le n-1$ except the intermediate goal and those
      which have appeared in Track-Tree into Candidate-Queue
      and go to Step 6.
Step5:Return that the intermediate goal can not be found.
Step6:End.
```

Procedure 2: (Backtrack_Track-Tree)

```
Step1:Move the robot to the position corresponding to the *nearest*
      ancestor node of the current node in the Track-Tree which is
      not a *dead node*.
Step2:Move the robot to the location corresponding to the element
      with the highest priority in the current Candidate-Queue and
      generate a node in Track-Tree.
Step3:Return.
Step4:End.
```

The fact that application of the navigation algorithm will lead the mobile robot from the source S to the goal G through a collision-free path is provided in the following theorem whose proof is given in [29].

Theorem 3.1: (Collision-Avoidance and Goal-Convergence)
Consider the navigation problem in section 2 satisfying assumptions **(A1)-(A3)**. The navigation algorithm given above will navigate the robot through a collision-free path from the source S to the goal G.

Remark : Such an algorithm is, in fact, performing a depth-first search over a tree composed of the source S, the goal G, and essentially the set of points each of which corresponds to the "middle" point of some pair of two confronted vertices of two close-by obstacles.

4. Simulation Examples

In this section, an examples is provided to illustrate the performance of the navigation algorithm presented in section 3.

Example :

Planning a collision-free path in the environment shown in Fig. 4.1 will require the use of procedure Backtrack_Track-Tree. Apparently, there may be some "misleading" in the first few stages due to the "U" type concave obstacle right in front of the goal G. This, hence, leads to the use of the procedure Backtrack_Track-Tree at position T. It can be seen that the order of the via points of the resultant path is S -> m_1 -> T -> m_1 -> S -> m_2 -> m_3 -> m_4 -> G. In this environment, the path does not violate the "center-line" criterion. But in some special situation, the resultant path will no longer be a center-line path. However the collision-free path always can be found if it exists.

5. Navigation Algorithm for Limited Distance Measurability of the Sensor Device

In section 3, we present an algorithm which assumes that the effective range of sensing of the sensor device is unlimited. But in practice, this will not hold in general. If that is the case, the algorithm just presented has to be modified to cope with the "near-sightedness" of the sensor, that is, although some vertices and edges are visible from the location of the robot, they cannot be detected by the sensor. Therefore, the information about the environment which has been collected and recorded onto Global-Map each time now plays a crucially important role in helping to determine the next immediate goal.

To be more specific, the difference of the ideal sensing from the practical sensing mainly lies in the fact that the clusters of visible vertices in the latter case are only subsets of those in the former case and may even be empty sets. Thus, one may have difficulty in performing the procedure Find_Intermediate_Goal since the obtaining of p^i or q^i of the cluster $V_i(p^i,q^i)$ as given previously out of the current sensor reading can no longer be assumed straightforward. To remedy this problem, two measures should be adopted: one is to steer the robot into a location where some obstacle begin to appear in the effective range of sensing so that data of vertices as well as edges can be collected and recorded; the other is to use the "memory" from Global-Map updated so far in order to recover $V_i(p^i,q^i)$, $i=1,\dots,n$ as much as possible so as to find out those intermediate goals. Before we make these measures algorithmic, two more definitions will be introduced, namely, *clusters of detectable vertices*, $D_i(\bar{p}^i,\bar{q}^i)$, $i=1,\dots,m$, and *clusters of critical vertices*, $C_j(\hat{p}^j,\hat{q}^j)$, $j=1,\dots,\bar{m}$, in the following:

Definition 5.1 : (Cluster of Detectable Vertices (CDV))
Since the distance measurability of the sensor is limited, not all the visible vertices from the location of the robot can be detected by the sensor. By grouping all the detectable vertices out of the sensor reading in a way similar to that in defining CVV, we obtain clusters of detectable vertices. This along with related edge information is then used to update Global-Map.

Definition 5.2 : (Cluster of Critical Vertices (CCV))
Global-Map is a disconnected graph as mentioned before. Every time after Global-Map is updated, the environment is reconstructed with obstacles now becoming walls which correspond to edges. Thus, all the visible vertices from the current robot location are then defined as clusters of critical vertices.

It is clear from the definitions that we have to find CDV before we can determine CCV at each iteration. The following procedure describes a method of obtaining CCV.

Procedure 3 : (Find_Clusters_of_Critical_Vertices)

```
Step1:Find CDV, $D_i(\bar{p}^i,\bar{q}^i)$, $i=1,\dots,m$.
Step2:Update Global-Map and reconstruct the environment.
Step3:Find CVV, $V_j(\hat{p}^j,\hat{q}^j)$, $j=1,\dots,\bar{m}$. Then, CCV is
      $C_j(\hat{p}^j,\hat{q}^j)=V_j(\hat{p}^j,\hat{q}^j)$, $j=1,\dots,\bar{m}$.
Step4:Order the clusters $C_j(\hat{p}^j,\hat{q}^j)$, $j=1,\dots,\bar{m}$, such that $\hat{q}^j$ and
      $\hat{p}^{j+1}$ are close by.
Step5:End.
```

This procedure will now replace Step 1 of the procedure Find_Intermediate_Goal introduced in section 3. In the foregoing observation, there exists a situation where the sensor on the robot cannot detect any further new information due to its limited distance measurability. It is clear from section 3 that the procedure Backtrack_Track-Tree can be invoked to resolve this problem if the ideal sensor is instead assumed. However, in the practical case given in this section, the continuous performance of the backtracking procedure may only lead the robot back to the starting point with no solution path found. The remedy is to gain more information from the environment so that Global-Map can be updated more completely. The following procedure will provide a systematic method in realizing this idea.

Procedure 4 : (Learn_More_Information)

```
Step1:Move the robot toward the goal direction with a displacement equal to the effective range of the sensor or till it reaches some edge of an obstacle, depending on which is satisfied earlier.
Step2:Add the current position of the robot into Track-Tree.
Step3:If the robot reaches some edge of an obstacle,
      then slide the robot along the edge in either direction till it reaches a vertex not visited before, and add the location data of the vertex into Track-Tree.
Step4:Scan the environment and add new information onto Global-Map.
Step5:If some new information is obtained,
      then go to Step 7;
      else go to Step 6.
Step6:If the robot is located on an edge of some obstacle,
      then slide the robot again along the inhabited edge, but not allowed to pass by any point which already appeared in Track-Tree, till it reaches another vertex not visited before, add the location data of this vertex into Track-Tree, and go to Step 4.
      else go to Step 1.
Step7:End.
```

Navigation Algorithm* :

```
Step1:If the goal is visible,
      then go to the goal directly and exit the algorithm.
Step2:Scan the environment and add the collected information onto Global-Map.
Step3:If no new information can be collected,
      then perform the procedure Learn_More_Information.
Step4:Perform the procedure Find_Intermediate_Goal.
Step5:If the next intermediate goal is found,
      then perform the procedure Guarded_Move and go to Step 6.
      else if the confronted concave region is perceived,
           then perform the procedure Back_Track-Tree, and go to Step 1.
           else perform the procedure Learn_More _Information and go to Step 4.
Step6:If the intermediate goal is successfully reached,
      then generate a node in Track-Tree that corresponds to the current location of the robot, remove the node from all Candidate_Queue's constructed so far, and go to Step 1.
      else perform the procedure Backtrack_Track-Tree.
Step7:Go to Step 1.
```

Note that the procedure Guarded_Move actually corresponds to the confirmation process mentioned in the previous observation at the beginning of this section. It is described in detail below:

Procedure 5 : (Guarded_Move)

```
Step1:If the intermediate goal is detectable and validated ( i.e. within the effective range of the sensor, and out of any obstacle or not blocked by any obstacle ),
      then go to the intermediate goal directly and go to Step 3;
      else go to Step 2.
Step2:If the intermediate goal is outside the range of effectiveness of the sensor and none of the obstacles lying between the current robot location and the intermediate goal are detected,
      then move the robot toward the goal direction with a displacement equal to the effective range of the sensor and generate a node in Track-Tree, corresponding to the current location and go to Step 1.
      else go to Step 3.
Step3:End.
```

Theorem 5.1 : (Collision-Avoidance and Goal-Convergence)

Consider the same problem as given in Theorem 3.1 but in the case where the sensor has only limited distance measurability. Then the navigation algorithm* proposed above will lead the robot from the source S to the goal G through a collision-free path.

The proof is provided in [29]. In the above algorithm, the robot is assumed to be volumeless. However, when the robot has non-zero volume, the same algorithm can still be applied using the method in [13].

6. Navigation through a Terrain Populated with Non-Polygonal Obstacles

In the previous sections, the navigation algorithms proposed only deal with polygonal obstacles. In this section, we will show that by only slight modification of the previous algorithm the similar results can also be applied to the case where obstacles may be non-polygonal. For ease of presentation, only the case where the sensor has unlimited effective range is considered here.

The idea, roughly speaking, is to gradually construct ever-improving approximating polygonal shapes for all the non-polygonal obstacles during the process of navigation so that the present environment asymptotically converges to the one encountered previously, i.e. an environment populated simply with polygonal obstacles. Clearly, if the construction is fine enough, a generic collision-free path in the original environment can still be found using the information from the new environment (containing polygonal substitutes).

Because the obstacle is non-polygonal, Definition 3.4 is no longer appropriate for recording the useful information observed by the robot sensor since there may not exist so called "vertices". In order to gain the similar function, we will first define the outwardmost point, say, p^i of some visible obstacle i as the intersection point of the line, emitting from the current robot location, tangent to the obstacle and the obstacle itself. If one of the two tangent lines is overlapping with some other obstacles, then the pertaining outwardmost point is then not defined. For illustration, in Fig. 6.1 there are two clusters of visible tangent points denoted, $T_1(p^1,p^1)$ and $T_2(p^2,q^2)$, and in Fig. 6.2 there are also two clusters of visible tangent points, $T_1(p^1,q^1)$ and $T_2(p^3,q^3)$ (although there are, in fact, three visible obstacles here). Whenever these visible tangents points are found, the generating tangent lines are also recorded appropriately by some data structure.

In addition, a maximal number of tangent lines (MNTL) is defined for the environment prior to the navigation process. If the number of tangent lines of an obstacle recorded so far is equal to MNTL, the tangent lines to the same obstacle constructed in the subsequent iterations are no longer recorded when it is observed again. From then on, that particular obstacle will be replaced by its polygonal substitute when performing the search for the next intermediate goal. With the replacement, the cluster of visible vertices can thus be similarly defined. Of course, it may not be necessary to construct MNTL tangent lines for every obstacle if we are only to find a collision-free path from the given source to the final goal. In Fig. 6.3, the most current polygonal substitute of the obstacle, namely, the polygon $p^1q^2q^1p^2$ is given by four tangent points. Conceivably, the larger the MNTL is, the more accurate the approximation is of a non-polygonal obstacle by a polygonal substitute. Therefore, after some obstacles are replaced by their final polygonal substitutes, the intermediate goal found in a way given in section 3 is more likely collision free for higher MNTL.

Now we are ready to state the navigation algorithm which particularly deals with the unknown environment clustered with non-polygonal obstacles.

Navigation Algorithm :**

```
Step1:Set the value of MNTL.
Step2:If the goal is visible,
    then go to the goal directly and exit the algorithm.
Step3:Scan the environment and add new information onto
    Global-Map.
Step4:If no obstacle can be observed because the robot already
    sits on the curved edge of an obstacle,
    then move the robot along the edge in either direction till it
    reaches a point from which more information can be
    observed, add the location data of this point into Track-Tree,
    and go to Step 2.
    else perform the procedure Find_Intermediate_Goal.
Step5:If the next intermediate goal is found,
    then go to Step 6;
    else perform the procedure Backtrack_Track-Tree and go to
    Step 2.
Step6:If the intermediate goal is outside any obstacle,
    then move the robot to that point directly, generate a node
    in Track-Tree, remove the node corresponding to the current
    location from all Candidate Queue's constructed so far, and
    go to Step 2.
    else perform the procedure Backtrack_Track-Tree and go to
    Step 2.
```

The procedure Find_Intermediate_Goal in the above algorithm is a modified version of the one introduced in section 3. The modification is made in its Step 2 and the current version is given below:

```
Step2:If the tangent lines recorded so far of some visible obstacle
    $i$ is equal to MNTL+1,
    then the newly found cluster of tangent points and the
    tangent lines of that obstacle are not recorded, and the
    polygonal substitute of the obstacle is used to determine the
    clusters of visible vertices $V_i(p^i, q^i)$,
    else the cluster of tangent points as well as the tangent lines
    are recorded.
    Create a Candidate-Queue.
```

The following theorem will assure us of the collision-free and goal-convergent properties. The proof of the theorem is provided in [29].

Theorem 6.1 : (Collision-Avoidance and Goal-Convergence)
Consider the problem set-up as given in Theorem 3.1 with non-polygonal obstacles. Then for sufficiently large MNTL the navigation algorithm proposed above will successfully navigate the robot through a collision-free path from the source S to the goal G.

7. Discussions

The algorithm proposed here bears similarity to that given by Rao et al [21] when only polygonal obstacles are dealt with and when the sensor possesses unlimited effective range. Under that circumstance, the update of Global-Map in the current work will be the same as the learning process in the work of [21]. However, the final solution path found there will consist of "vertex-to-vertex" segments as opposed to the center-line path attempted here. Furthermore, the aim of finding a better path in [21] after the terrain information has been learned enough is also different from the theme of this paper which is simply to find one of possibly several solution paths. Despite these quite comparable results, this current work can handle more general situations than those in [21] such as the environment with non-polygonal obstacles and practical limitation of the sensor. Notably, the investigation of the latter is so far original to the author's point of view.

8. Conclusions

In this research, an algorithm which navigates a mobile robot through an unknown terrain populated with obstacles of general shapes was presented. The algorithm tends to find a center-line collision-free path, which is recognized as a safer scheme in the robotics literature. An original investigation of the case where the robot sensor possesses only limited distance measurability was also given in this paper. To guarantee the property of collision-avoidance and goal-convergence of the algorithm, several rigorous proofs were provided. A few computer simulation examples were given to illustrate the performance of the algorithm.

References:

[1] R. Bhatt, D. Gaw, and A. Meystel, "A Real-Time Guidance System for an Autonomous Vehicle," Proc. 26th Conf. Decision and Control, pp 1785-1791, 1987.

[2] R.A. Brook, "Solving the Find-Path Problem by Good Representation of Free-Space," IEEE Trans. Systems, Man, and Cybernetics, Vol. SMC-13, No. 3, 1983.

[3] R.A. Brook and T. Lozano-Perez, "A Subdivision Algorithm in Configuration Space for Find-Path with Rotation," IEEE Trans. Systems, Man, and Cybemetics, Vol. SMC-15, No. 2, pp 224-244, 1985.

[4] R. Chattery, "Some Heuristics for the Navigation for a Robot," Int. J. Robotics Research, Vol. 4, No. 1, pp 59-66, 1985.

[5] J.L. Crowley, "Navigation of an Intelligent Mobile Robot," IEEE J. Robotics and Automation, RA-1, Vol. 2, pp 31-41, 1985.

[6] A. Elfes, "Sonar-Based Real-World Mapping and Navigation," IEEE J. Robotics and Automation, RA-3, Vol. 3, pp 249-265, 1987.

[7] G. Giralt, R. Sobek, and R. Chatila, "A Multlevel Planning and Navigation System for a Mobile Robot," Proc. Int. Joint Conference on Artificial Intelligence, pp 335-338, 1979.

[8] L. Gouzenes, "Strategies for Solving Collision-Free Trajectory Problems for Mobile and Manipulator Robots," Int. J. Robotics Research, Vol. 3, No. 4, pp 51-65, 1984.

[9] R. Hoffman and A.K. Jain, "Segmentation and Classification of Range Images," IEEE Trans. Pattern Analysis and Machine Intelligence, VOL. PAMI-9, NO. 5, 1987.

[10] S.S. Iyengar, C.C. Jorgensen, S.V.N. Rao, and C.R. Weisbin, "Robot Navigation Algorithms Using Learned Spatial Graphs," Robotica (1986) Vol. 4, pp 93-100.

[11] C.H. Lin and L.C. Fu, "A Center-Line Oriented Robot Navigation in an Unknown Terrain," Proc. of International Computer Symposium, Taipei, Taiwan, Republic of China, pp 235-240, Dec. 1988.

[12] T. Lozano-Perez, "Spatial Planning: a Configuration Space Approach,"

IEEE Trans. Computers, Vol. C-32, No. 2, pp 108-120, 1983.

[13] T. Lozano-Perez and M.A. Wesley, "An Algorithm for Planning Collision-Free Paths among Polyhedral Obstacles," Communications, ACM, Vol. 22, pp 560-570, 1979.

[14] L. Matthies, and S.A. Shafer, "Error Modeling in Stereo Navigation," IEEE J. Robotics and Automation, VOL. RA-3, NO. 3, pp 239-248, 1987.

[15] M. Montgomery, D. Gaw, and A. Meystel, "Navigation Algorithm for a Nested Hierarchical System of Robot Path Planning Among Polyhedral Obstacles," Proc. 26th Conference on Decision and Control, pp 1616-1622, 1987.

[16] H.P. Moravec, "The Stanford Cart and the CMU Rover," Proc. of the IEEE, VOL. 71, NO. 7, pp 872-884, July, 1987.

[17] N.J. Nilsson, "A Mobile Automation: An Application of Artificial Intelligence Techniques," Proc. Int. Joint Conf. Artificial Intelligence, pp 509-520, 1969.

[18] J.L. Olivier and F. Ozguner, "A Navigation Algorithm for an Intelligent Vehicle with a Lazer Range-Finder," Proc. IEEE Int. Conf. Robotics and Automation, pp 1145-1150, 1986.

[19] B.J. Oommen, S.S. Iyengar, and N.S.V. Rao, and R.L. Kashyap, "Robot Navigation in Unknown Terrains Using Learned Visibility Graphs, Part I: The Disjoint Convex Obstacle Case," IEEE J. Robotics and Automation, VOL. RA-3, NO. 6, pp 672-681, 1987.

[20] S.V.N. Rao, S.S. Iyengar, C.C. Jorgensen, and C.R. Weisbin, "Robot Navigation in Unexplored Terrain," Journal of Robotic Systems, 3(4), pp 389-407, 1986.

[21] S.V.N. Rao, S.S. Iyengar, B.J. Oommen, and R.L. Kashyap, "On Terrain Model Acquisition by a Point Robot Amidst Polyhedral Obstacles," IEEE J. of Robotics and Automation, VOL. 4, NO. 4, pp 450-455, 1988.

[22] S.H. Suh and K.G. Shin, "A Variational Dynamic Programming Approach to Robot-Path Planning With a Distance-Safety Criterion," IEEE J. Robotics and Automation, VOL. 4, NO. 3, pp 334-349, June 1988.

[23] A.M. Thompson, "The Navigation System of the JPL Robot," Proc. Int. Joint Conference on Artificial Intelligence, pp 749-757, 1977.

[24] C.E. Thorpe, "Path Relaxation: Path Planning for Mobile Robot," Proc. AAAI, 5th Nation Conference on Artificial Intelligence, pp 318-321, 1984.

[25] S. Tsuji, J.Y. Zheng, and M. Asada, "Stereo Vision of a Mobile Robot: World Constraints for Image Matching and Interpretation," Proc. IEEE Int. Conf. Robotics and Automation, pp 1594-1599, 1986.

[26] S.M. Udupa, "Collision Detection and Avoidance in Computer Controlled Manipulators," Proc. 5th Int. Joint Conf. Artificial Intelligence, MIT, Cambridge, MA, pp 737-748, 1977.

[27] A.M. Waxman, J.L. Moigne, L.S. Davis, E. Liang, and T. Siddalingaiah, "A Visual Navigation System," Proc. IEEE Int. Conf. Robotics and Automation, pp 1600-1606, 1986.

[28] C.R. Weisbin, J. Barhen, G.de Saussure, W.R. Hamel, C.C. Jorgensen, E.M. Oblow, and R.E. Ricks, "Machine Intelligence for Robotics Applications," Proc. Conf. Intelligent Systems and Machines, pp 47-57, 1985.

[29] C.H. Lin and L.C. Fu, "Robot Navigation through Obstacles of General Shapes Using a Center-Line Oriented Algorithm," Technical Report NTUCSIE 89-10, Department of Computer Science & Information Engineering, National Taiwan University, Taipei, Taiwan, R.O.C.

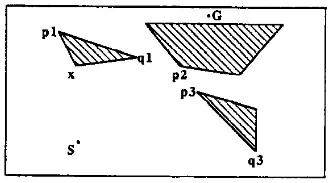

Fig. 2.1

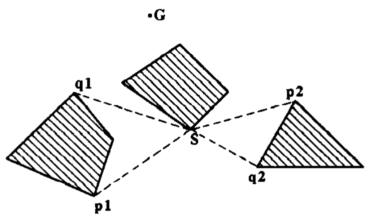

Fig. 3.1 The robot is on the vertex of obstacle

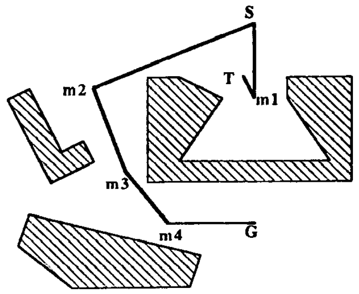

Fig. 4.4

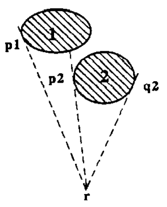

Fig. 6.1

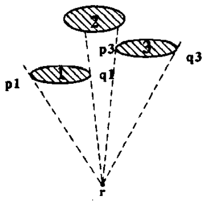

Fig. 6.2

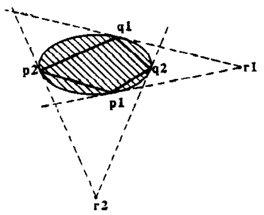

Fig. 6.3