A NEW 2-D WORLD REPRESENTATION SYSTEM FOR MOBILE ROBOTS

Qian Chen, Minoru Asada and Saburo Tsuji

Department of Control Engineering, Osaka University
Toyonaka, Osaka, 560 Japan

ABSTRACT

When a robot acquires new data about the its environment, it updates a collision-free space. Conventional methods suffer from reconstructing the free-map for each updating.

The method proposed in this paper divides the free space into convex polygons which can be considered as "path segments". A path segment is generated for each pair of free edges on the same polygon. Then the path segments sharing a common free edge will be linked. This leads to a connective graph indicating the collision free paths between the free edges.

When new data about the world are obtained, we only need to update the convex polygons and their path segments on which new information is available, and modify the links connecting these path segments to others.

1. INTRODUCTION

Path planning for an autonomous mobile robot needs a compact world representation which has been extensively studied, especially for indoor environments[1-5]. A polygonal representation of 2-D world proposed by Chatila and Laumand[2], characterized by its compactness, is useful for path planning. Another advantage of it is that when the robot acquires new data on obstacles from sensory information, the model can be updated easily. Modification of the model is limited to the polygons on which new information is available, and the other parts of models are left unchanged. The method, however, neglects the robot's size and, therefore, it cannot guarantee the collision-free paths in convex polygons unless the polygons are much larger than the robot.

One idea to solve this problem is that we build a free map from the polygon representation by expanding the obstacle regions so as to consider the robot as a point[1]. But the updating of this model is inefficient because the effect of new world data propagates over the free map, and we must reconstruct significantly large portion of the model.

In order to overcome these difficulties, we propose a new polygon-based representation of 2-D world. At first, the world is divided into convex polygons, and then we examine existence a "path segment" through which the robot moves from each free edge of a polygon to the other free edges. Thus, the polygons are represented by a graph where nodes are free edges and arcs are the path segments. Integration of these graphs yields a connecting graph of the world which is useful for path planning.

2. PROBLEMS ABOUT THE PREVIOUS MODEL

Polygon-based 2-D space models have been studied in the research on path planning. Here, we will discuss problems in updating these models when new data of the environment are acquired. ne kind of the models is free space model, in which the robot can be considered as a single reference point. This model (it will be hereinafter referred as "free map") can be obtained by expanding the obstacle regions according to the shape of the robot. Here, we assume that the shape of the robot is a disk.

The edges of the free map consist of line segments generated by expanding obstacle regions and arcs at concave corners of free space. For the convenience of path planning, we approximate the arcs by line segments. When some edges of the free space change, we will not be able to predicate which part of free map need to be updated and which part need not. This is because there is no one-to-one relation between the edges of the map and those of the free space. Some examples are shown in Fig. 1 So each time the world changes, the free space has to be reconstructed.

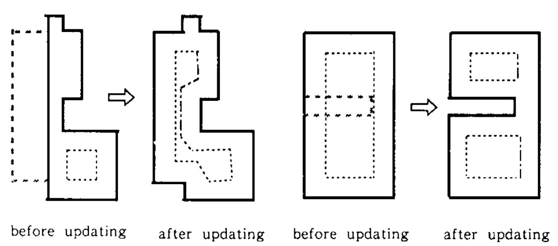
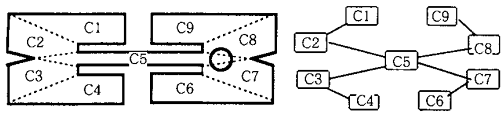


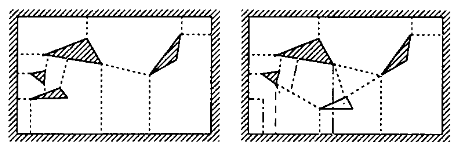
Fig. 1 Two examples of updating of free map. Notice that some edges of free map are generated from unchanged obstacle edges, and some change that their corresponding obstacle edges remain unchanged, marked as -·-·-·-·-

Another approach is to represent the free space in terms of basic shape primitives. Chatila's approach is to decompose the free space into convex polygon cells, and building a connective graph where nodes are the cells and arcs are the links corresponding to the common edge segments shared by adjacent cells. When the free space changes, only the cells that the edges of them changed need to be updated. But the method does not consider the size of robot, so it gives incorrect paths sometimes. (shown in Fig. 2) Unlike Chatila's approach, Kuan decomposes the free space by cutting it with edges called "minimum distance". The minimum distance is the shortest edge of all vertex−to−vertex, or vertex−to−edge distances between two obstacles. This also leads to a connective graph. When some obstacles changes, not only the minimum distances corresponding to it will change but some new minimum distances will be generated from some unchanged obstacles. An example is shown in Fig. 3. Therefore, we need to reconstruct the model when some obstacles changed.



robot Ci (i=1,2,...) : convex polygon cell

Fig. 2 A descomposed concave free space and its connective graph. Note that C1 and C9 are connective, but the robot can not go from C1 to C9.



obstacle changed obstacle

minimum distance

changed minimum distance of the changed obstacle

changed minimum distance of the unchanged obstacle

new minimum distance generated from unchanged obstacle

Fig. 3 Updating of minimum distances

Generally, each time new data about the free space comes, most of the free space remains unchanged and so of the free map. If the free map is reconstructed each time when updating, the cost of updating will seems very high.

3. OUR APPROACH

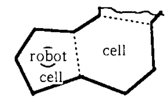
A good world representation for a mobile robot should have the following properties :

1. It is compact ;
2. It describes the world correctly ;
3. It is easy to use for path planning ;

and, 4. It can be updated easily.

Chatila's model satisfies **1** and **4**, but it yields incorrect paths when the convex polygons are not large enough. Here we try to build a new world model with both good properties of Chatila's model, and capability for finding collision−free paths.

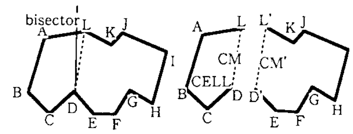
By decomposing the floor space, we obtain a set of convex polygon cells. A convex polygon cell has two kind of edges ; one kind is of the obstacle edges and the other is of those added by decomposing the floor space. Here, we define the latter edges as the **gates** of the cell. If the robot can go through a cell from one of its gates to another without collision against its obstacle edges, then we say there exists a path segment for the robot between the two gates, and it is represented by a link connecting the two gates. In order to construct a connective graph to represent the possible paths in the environment, we first find out all the path segments of each cell, and then link each two path segments of adjacent cells sharing a gate. (The condition for linking of the two path segments will be mentioned later.) Thus, a graph called a path graph is built, which is then used for path planning. (See Fig. 4)



obstacle edges

GATEs

Fig. 4. Convex polygon cells and GATEs.



CELL= ((A, B, C, D, L), (CM1))

CM1= (CM, CM')

Fig. 5. Decomposing concave polygon.

3.1 Decomposing Polygonal Free Space

The floor space can be represented with polygons. The polygonal free space may be convex or concave, furthermore, it may have holes (obstacles) in it.

Convex polygons without holes have a good property that makes path planning easier. For example, if a convex polygon has two edges such that the robot (assumed to be a disk) can go across them without collision against other edges, then the robot can go through the polygon without collision. In order to make use of this property, the free space is decomposed into convex polygon cells.

The decomposition algorithm finds a concave angle in the polygon and then finds another vertex of the polygon such that the segment connecting them is in the polygon and has the minimum angle to the bisector of the concave angle. Then the polygon is decomposed into two by adding a pair of the segments, as shown in Fig. 5. This algorithm is recursively applied until there exists no concave corner.

The result of this algorithm is a set of convex polygon cells. Each cell is represented as the form (C, (CM1, CM2, CM3, ...)), where **C** is the cell itself and **CMi** (i=1,2,...) are the common edges shared by the cell and its adjacent cells.

3.2 Path Segment

First, we examine path segments of each cell without considering the influences from other cells. If a cell has more than two gates, we examine all pairs of them ; for each pair, all other edges are regarded as obstacle edges which will be called as **walls** hereafter.(See Fig. 6) Obviously, the robot cannot enter a gate if it is narrower than robot's width. We regard gates of this kind as obstacle edges. Even if the gates are wide enough, the robot sometimes still cannot go through the cell because of collision with obstacle edges of the cell. To judge whether the robot can go through a cell, we first make the following definitions.

Definition 1 :

The robot should not collide with the walls while going through the cell. For each wall, as shown in Fig. 7, we draw a group of edges apart from the wall by the robot's radius R. We call them **safesides**. The robot does not collide with the walls if its center does not enter in between a safeside and its corresponding wall.

Definition 2 :

In order for to enter the cell through the gate without colliding against the walls, the center of the robot must be kept off both walls farther than its radius R. If any point on the gate does not satisfy the condition, we consider the gate is **closed**, and regard it as an obstacle edge. If a range on the gate satisfies the condition, the gate is said **open** and, wel register the range as the **door** in a data structure representing each cell (Fig. 8).

If both of the two gates are open, the robot will be able to go through the cell without collision against obstacle edges, and we connect the two gates with a link representing a path segment.

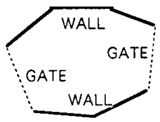


Fig. 6. WALLS and GATES

3.3 Path Graph Construction

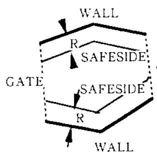


Fig. 7. Safeside.

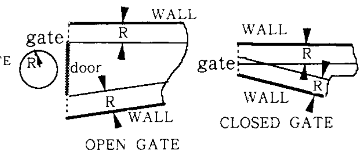


Fig. 8. OPEN GATE and CLOSED GATE.

A convex polygon cell and its adjacent cell share a common edge. It is the **gate** as we defined in the beginning of this chapter. Thus, two path segments belonging to a polygon cell and its neighbor share a common gate. Note that the **door** of a gate indicates the range that robot can go through. So if two doors of the common gate (belonging to the two path segments) overlap, then the robot can go across the gate from one path segment to the other. In this case, we link the " two" common gates with the overlapping part of the two doors as shown in Fig. 9. This leads to a connective graph.

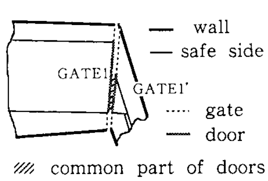


Fig. 9. Link the two gates

4. PATH PLANNING

Given the start and goal points (the location of the robot is represented by its center), we first find out which cell contains the start (goal) point, and, we call it the initial (goal) cell. Then we check whether the robot is colliding with obstacle edges at its initial (goal) point. If so, the navigation is impossible. Otherwise, we find out all the gates of the initial (goal) cell and call them initial (goal) gate. Then the path planning problem becomes search the path-graph for a path from one of the initial gates to one of the goal gates. Many strategies can be used to find a path.

5. MAP UPDATING

Like Chatila's, our model can be updated efficiently when the robot acquires new data about the world. When some obstacle edges change, we only have to update the cells that some of their obstacle edges are of those changed, the gates on those cells and the link that connect them to those of the unchanged cells.

6. Experiment and Result

We have built a simulation system of a mobile robot working like a map-maker with the model proposed. The robot is put in a indoor room, and its job is to find the free space with its stereo vision system where the robot can move about, and build a map to represent the free space. Given a initial position and a goal position, the system can find out a collision free path with the map.

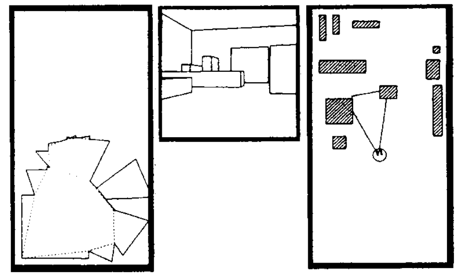


Fig. 10a. Map before updating, a new observation, and a newly found floor space.

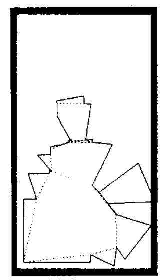


Fig. 10b. Map after updating.

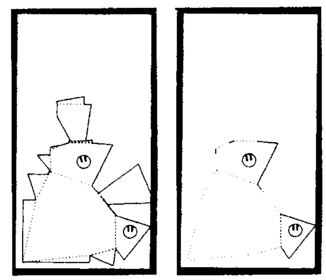


Fig. 11. A initial position and a goal in the map, and the path found between initial positionand goal

7. CONCLUSIONS

Efficient updating of world model for mobile robots is important, because the robot frequently iterates move-and-view operations. The 2-D world representation described in this paper can be updated efficiently because it represents the world in terms of its primitives (convex polygon cells), and it is good for path planning because the paths from one place to another are described explicitly in terms of gates and path segments.

REFERENCE

[1] Rodney a. Brooks, "Solving the Find-Path Problem by Good Representation of Free Space," IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC-13. No. 3. March/April 1983, pp. 190-197

[2] Raja Chatila and Jean-Paul Laumond, "Position Referencing and Consistent World Modeling for Mobile Robots," Proc. of IEEE International Conference on Robotics and Automation, pp 138-145, 1985.

[3] David Miller, "A Spatial Representation System for Mobile Robots," Proc. of IEEE International Conference on Robotics and Automation pp. 122-127, 1985.

[4] Darwin T. Kuan "Natural Decomposition of Free Space for Path Planning," Proc. of IEEE International Conference on Robotics and Automation, pp 68-173, 1985.

[5] James L. Crowley "Navigation for an Intelligent Mobile Robot,"IEEE Journal of Robotics and Automation, Vol. RA-1, No.1, March 1985. pp. 31-41

[6] John H.Reif"Minimizing Turns for Discrete Movement in the Interior of a Polygon," IEEE Journal of Robotics and Automation, Vol. RA-3, No. 3, June 1987, pp 182-193.