Path Planning for Mobile Robot Based on Improved Framed-Quadtree*

Chang'an Liu, Wei Liu, Hong Zhou, and Zhenhua Wei

School of Computer Science and Technology, North China Electric Power University
Beijing, 102206 China
liuanhit@163.com

Abstract. A path planning method based on improved Framed-Quadtree is researched for mobile robot in this paper. Firstly, this paper analyzes drawbacks of the quadtree method and according to those, proposes the method which uses the framed-quadtree to model the search space. This method avoids the condition which is difficult to find the shortest path sometimes. Secondly, the storage structure of the framed-quadtree is improved by using pattern code method. Therefore, it saves the original storage space, improves the environment resolution and reduces the search time. Finally, the optimal path is found by using A* algorithm. The experimental results prove the correctness and validity of the method.

Keywords: mobile robot, path planning, framed-quadtree, pattern code.

1 Introduction

The problem of finding optimal path for a mobile robot means to find a collision-free and the shortest distance path which is from the given point to the end point in the clutter environment with obstacles. The robot can avoid all obstacles and track the path securely and freely. Path planning manifests the independent ability of intelligent mobile robot. It has two essential sorts in [1] [2]: One is global path planning based on model that the environment of the robot is certain and the other is local path planning based on sensor that the environment of the robot is uncertain. The global path planning includes generalized taper method and grid method, the local path planning includes artificial potential field theory described in[3] [4]. All of these methods have their advantages and disadvantages. In these methods, Quadtree described in [5] is perfect.

Taking into consideration the disadvantages of quadtree that in some cases, the quadtree method can not find the optimal path, this paper models the environment using framed-quadtree and improves its storage structure. At last path planning is implemented by A* algorithm. The result shows that it is better to reach the goal and more effective than before.

* This work is partially supported by the Nationl Science Foundation of China (60775058) and the Key Project of China Ministry of Education(107028).

2 The Configuration of Environment Space

The first step in path planning is to establish environment model properly. The familiar methods include configuration algorithm, corner image method, grid method, image method, quadtree method and so on. Quadtree modeling based on encoder is a perfect modeling method. The familiar methods for path planning are A* algorithm, genetic algorithm. Taking into consideration the disadvantages of quadtree, this paper model the environment using framed-quadtree and improve its storage structure.

2.1 Quadtree and Its Drawback

Reference [6] describes the quadtree structure which is a recursive decomposition of a two-dimensional environment into uniformly sized regions. The nodes of the quadtree are classified as either free nodes representing obstacle-free areas, obstacle nodes representing regions containing obstacles and which the robot cannot travel through, or mixed nodes representing regions of both obstacle and free space. Each mixed node must be recursively sub-divided into four sub-quads which form the children of that node. The Fig 1 shows the obstacles environment space, and the Fig 2 shows the environment after decomposing by quadtree method.

Current path planning approach using quadtree has drawback that the quadtree method cannot guarantee finding the shortest path. Since the position of a path graph

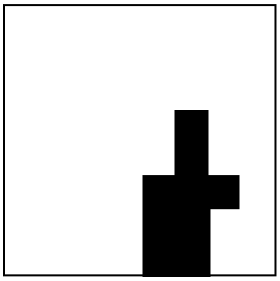

Fig. 1. A sample environment with obstacle

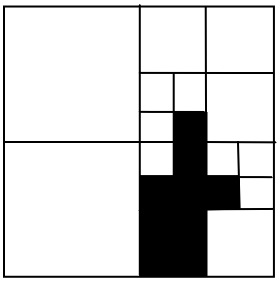

Fig. 2. The decomposition of the environment shown in Fig 1 using quadtree method

node is always the center of the corresponding cell, the cost of traversing the cell is the same whether the path just clips a corner of the cell or actually passes through the entire width of the cell. Hence, the quadtree is very sensitive to obstacle placement and the error factor is proportional to the size of the cell.

For example, Fig 3 describes a situation which plans a path starting from S to G using the quadtree. Since the paths generated using the quadtree should pass the center of each cell, path A and path C are found, the price of path C is lager than that of path A, so the path A is found at last by using Dijkstra's shortest path algorithm. While path B cannot be found that is obvious the desired shortest path.

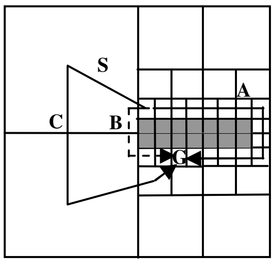

Fig. 3. The optimal path B can not be found by using quadtree approach

2.2 Framed-Quadtree

The drawback of the quadtree can be solved by introducing a new data structure, which is called framed-quadtree [7] [8]. The border of a large cell in a free space is enclosed by an array of square cells (called a "frame") which are of the smallest possible size allowed for a particular environment. Then the primary node is cancelled. The node formed frame calls frame-node. Obviously, framed-quadtree not only has advantages of quadtree, but also can express environment more efficient than quadtree. Fig 4 shows the optimal path based on the framed-quadtree. Framed-node is the smallest unit. This way allows the path to go forward in many orientations, and can find the optimal path.

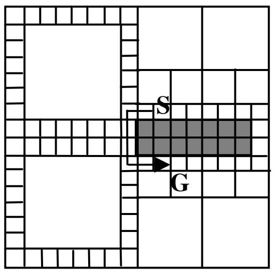

Fig. 4. Framed-quadtree

3 Storage Mode of Framed-Quadtree

3.1 The Traditional Storage Mode for Framed-Quadtree

Framed-quadtree structure as in [9] is a layered tree based on conformance criterion of data area and theory of recursion decomposition space. The conformance criterion of data area is that: with regard to the binary image, if a certain region of all the pixels are the same value (white or black), and the region is conform. The theory of recursion decomposition space is that each binary image will be divided into four district of the same size. If a district from all pixels is in the same color (black), then claim this district as black area. If a district from all pixels is in the same color (white), then claim this district as white area. If a district both from the black pixels and white pixels, then claim this district as gray area. The black and white area is no longer decomposed. The gray area will continue to be decomposed until to the required size. The decomposition can be represented by a quadtree, whose root node represents the entire image, and leaf nodes represent the black and the white area, while non-leaf nodes represent gray area. Nodes corresponding black, white and gray area are respectively called black node, white node and gray node. Fig 5 shows the storage mode of Fig 2 environment, using the basic framed-quadtree method.

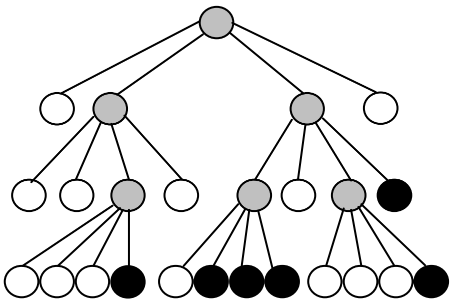

Fig. 5. Storage mode of Fig 2 by using framed-quadtree method

3.2 The Improved Storage Mode for Framed-Quadtree

Avoid combining SI and CGS units, Framed-quadtree storage is an important step, which is to improve space-time efficiency of environmental model. A record with five fields is used to represent each node of the tree in traditional framed-quadtree. In these five fields, one field is used to describe the characters of nodes (gray, white, black three types of nodes), and the other four nodes are used to store pointers pointing to its four sub-nodes. This is the commonly used method to express the tree storage structure and it is also the earliest method to be used in graphics and image processing. It has much drawback, the biggest one is that the pointers need a lot of space, it account for 90% of all storage space, so the utilization rate of storage space is very low. Using the pattern framed-quadtree (Pattern Framed Quadtree Trees, PFQT) can reduce the number of nodes in the framed-quadtree, to save occupied space of framed-quadtree and speed up processing.

PFQT express the binary image by using a series of pre-defined images whose size is variable as a node leaf. Four sub-districts after districted are the combination of black and white sub-districts. These patterns using the combination need less leaves than before, reduce the height of framed-quadtree, and achieve the purpose of saving storage space. Fig 6 illustrates 16 kinds of patterns in the PFQT.

The size of leaf node is consistent with the size of grid in path planning. The size of grid directly impacts on the performance of path planning algorithm. If grid size is small, then the environment resolution is high, but the capability of anti-interference is weak, the storage capacity of environment information is large, the speed of decision-making is slow. If grid size is large, then the environment resolution is weak, but the capability of anti-interference is strong, the storage capacity of environment information storage is small, the speed of decision-making is fast and the ability of finding obstructions in environment is weak. To cooperate with the framed-quadtree algorithm, the grid size is selected as half of the minimum width which is the robot can through (when obstacles dense environment, it is desired for 1/4 width of the autonomous robots).The stored mode of Fig 5 is shown in Fig 7 using PFQT.

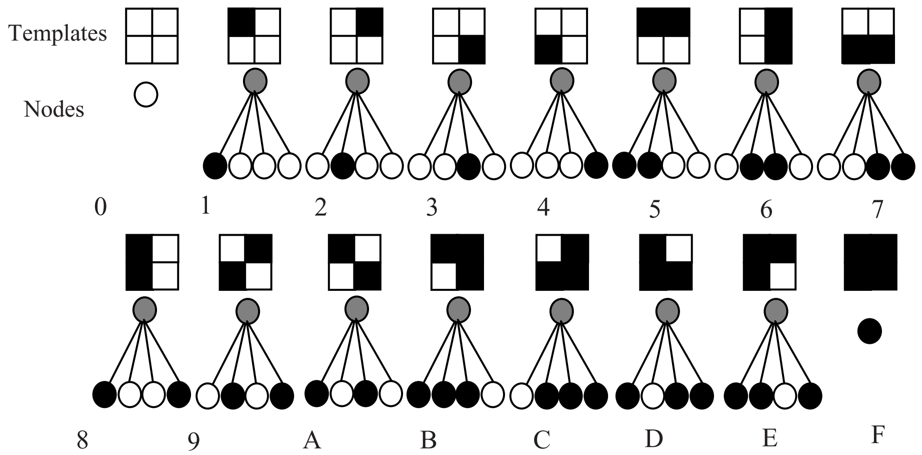

Fig. 6. PFQT and pattern coder

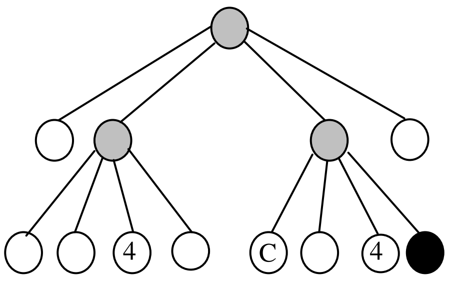

Fig. 7. Storage mode of fig 5 by using PFQT

4 PFQT Path Search Method

Heuristic search algorithm is optimized, which navigate by heuristic function. The search algorithm based on the map search algorithm, adds the heuristic function-value, which determines the expansion order of nodes. Heuristic search evaluates each search position in the state space, and finds the best position, then searches it until finding the search target. This can save a large number of unnecessary paths, also improve efficiency. In heuristic search, it is very important to evaluate the location. Use different evaluation function may have different effects. A* algorithm is a kind of heuristic search algorithm as in [10] in artificial intelligence field. The valuation in heuristic search is expressed by evaluation function. The selection of evaluation function is directly related to the success or not of A* algorithm, however, the determination of the evaluation function has a close relationship with actual situation. So it is important to choose evaluation function.

4.1 The Selection of Evaluation Function

The core of A* algorithm is evaluation function $f(n)$, which includes $g(n)$ and $h(n)$.

$$f(n) = g(n) + h(n) \tag{1}$$

$g(n)$ is the value which has been traversed and $h(n)$ is the value which is from node n to the target node, $g(n)$ shows the depth from the current node to n in the state space, and $h(n)$ is the distance from the location of node n in the map to the target node.

$$h(n) = \sqrt{(dx - sx)^2 + (dy - sy)^2} \tag{2}$$

The selected grid size is 1×1 pixel, so $h(n) >= 1$ or $h(n) = 0$. In order to calculate conveniently, the assessed-value of node is $h(n)$'s square. Proportion is right, it avoids the calculation of prescribing and decimal, makes the calculation faster and more efficient. The obstacle value should be given a special value, which is not get by formulation of $h(n)$. The assessed-value function of obstacle grid here is 2000.

4.2 Major Steps of Algorithm

There are two queues in A* algorithm: the OPEN queue and the CLOSE queue. All nodes that have not visited are preserved in the OPEN queue, they are from their father nodes and arranged according to the evaluation value, so the first node is the smallest one.

There is only one node waiting to be visited in the queue; visited nodes are recorded by CLOSED queue. It probes other nodes from one node, if the node can be moved and this is the best way from the starting node to the current node, then it will be inserted into OPEN queue according to the assessed value. After probing all the neighbor nodes, the smallest node is putted into the CLOSED queue. In the same way it will probe the neighbor nodes of the smallest node, and add them to the OPEN queue.

The conditions adding to the OPEN queue are as follows:

(1). This step can move on the map;

(2). Nodes in this step does not exist in the OPEN queue.

(3). The real distance form the starting point to the current node is shorter than the history distance of this node.

If the three conditions are satisfied, then adding the current node into the OPEN queue. A* algorithm is a heuristic search algorithm based on evaluation function f (n).

The steps are as follows:

Step 1: Add the starting node into the OPEN queue;

Step 2: Circulate following steps:

(1). Travel OPEN queue, and look for the node of smallest value (assessed-value), which is treated as the node to be dealt with;

(2). Move this node into the CLOSE queue;

(3). For each neighbor box adjoined with current box, if it can not be arrived or is not in the CLOSED queue, then it will be ignored. Else follow the steps below:

a. If this node is not in the OPEN queue, it is inserted into the OPEN queue according to the order from small to large, and the current box is set to its father node. The f, g and h value of this box is recorded.

b. If it is already in the OPEN queue, then check this path (from current box to where it arrival) is better or not with g value for reference. The smaller value of g shows that this is a better path. If it is the better path, its father node is set to current node, and the value of g、 f is recalculated.

(4). If encounter the following situation, then stop searching:

a. If the target node is in the OPEN queue, then the final path is found already.

b. If the finding target node is failure, and the OPEN queue is empty, then the path does not exist.

Step 3: Preserve path. All the nodes in the CLOSED queue linked up form the final path.

5 Experimental Results

In order to demonstrate the feasibility, efficiency and the accuracy of the algorithm described in this paper, the experiment is implemented by Matlab. In this test environment, 100 points randomly generated simulate 100 obstacles. The smallest selected cell is 1×1 pixels. The environment model is established by framed-quadtree method. The storage of nodes is made by improved storage model. At last, the optimal path is shown in Fig 8 with A* algorithm. In this figure S is the start point and the G is the end point. According to the experimental results, the method in this paper can find the optimal path under given space constraints.

In order to demonstrate whether the method improved efficiency or not, two methods are compared in experiment. Table 1 show the comparison of 10 times planning

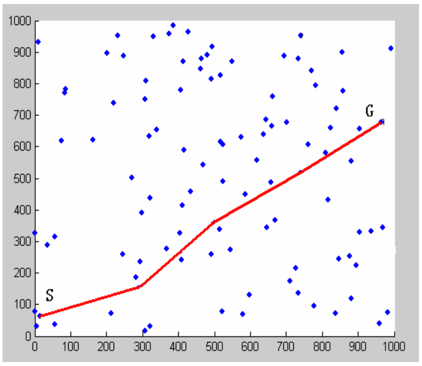

Fig. 8. Experiment result

Table 1. Experiment Results

	Basic method(s)	Improved method(s)
Optimal value:	0.27	0.11
Average value:	0.8	0.65

results in the same environment. The optimal value for the two methods is the computing time that is the fastest in 10 experiments, and the average value is the average time of 10 experiments.

In table 1, the improved method is more efficient than the basic method, because the node storage structure of the improved method is simpler than before. So, the access time is saved.

6 Conclusions

Improved framed-quadtree method is built on framed-quadtree, which improves the storage structure of framed-quadtree using model code. This method reduces the storage quantity of information and also in a certain extent, relieves the contradiction of the grid method between the environment resolution and the large quantity storage of environment information. This method improves the environment resolution and speeds up the algorithm processing. The experiment results prove that the method in this paper can resolve the path planning problem well.

References

1. Wu, T.: Research on Obstacle Avoidance and Path Planning for Mobile Robots. Hua Zhong University of Science and Technology, Wu Han (2004)
2. Wang, L.: Research on Path Planning for Mobile Robots. Harbin Engineering University, Harbin (2007)

3. Mech, E.D., Pamukkale, U., Denizli, et al.: Path planning using potential fields for highly redundant manipulators. Robotics and Autonomous Systems 52(2), 209–228 (2005)
4. Xu, X.Y., Xie, J., Xie, K.: Path Planning and Obstacle-Avoidance for Soccer Robot Based on Artificial Potential Field and Genetic Algorithm. In: Proceedings of the 6th World Congress on Intelligent Control and Automation, Dalian, pp. 21–23 (June 2006)
5. Zelinsky, A.: A mobile robot exploration algorithm. IEEE Transactions on Robotics and Automation 8(6), 707–717 (1992)
6. Li, H.C., Huang, Y.L., Que, J.L.: An Approach to Path Smoothing For wheeled Mobile Robots Based on Quadtree Environment Model. Robot 23(5), 426–430 (2001)
7. Chen, D.Z., Szczerba, R.J., Uhran Jr., J.J.: Planning Conditional Shortest Paths through an Unknown Environment: A Framed-Quadtree Approach. In: International Conference on Intelligent Robots and System, Pittsburgh, vol. 3, pp. 33–38 (August 1995)
8. Szczerba, R.J., Chen, D.Z., Uhran Jr., J.J.: Planning conditional shortest paths in an unknown environment. In: Proceedings of the Thirty-Second Annual Allerton Conference on Communication, Control and Computing, Illinois, pp. 671–672 (September 1994)
9. Yahja, A., Stentz, A., Singh, S.: Framed-quadtree path planning for mobile robots operating in sparse environments. In: Proceedings of the IEEE International Conference on Robotics and Automation, Belgium, vol. l(1), pp. 650–655 (May 1998)
10. Tang, P., Yang, Y.: Study on algorithm A* of intelligent path planning based on method of representation environment with both quad tree and binary tree. Control Theory and Applications 20, 770–772 (2003)