An Extension of Dynamic Programming Algorithm in Robotic Path Planning

Shanshan Ji,Lianhe Yang
School of Computer Science and Software, Tianjin Polytechnic University
Tianjin, China
710905035@qq.com, yanglh@tjpe.edu.cn

***Abstract*—DP (Dynamic Programming) algorithm based on distance-propagating is a very good approach to real-time collision avoidance in robotic path planning. But the problem with DP is the waste of step number of walking in tracking process. For the A* (A Star) algorithm, there are commonly three heuristic, and usually three ways (4-adjacency, 8-adjacency and 16-adjacency) to get the adjacencies. After The comparison and analysis of three ways in three heuristic, we find that the 16-adjacency is more suited to the DP algorithm. The 16-adjacency can expand the search area and then the extended DP algorithm will improve the search speed. Based on this, we extend DP algorithm called Extended DP. We can guarantee its completeness and optimality. On the experimental side, we demonstrate several simulations to illustrate its effectiveness.**

Keywords- robotic path planning, Dynamic Programming (DP), A star, Adjacency, Extended DP

I. INTRODUCTION

Path finding has been a traditional aspect of Artificial Intelligence for a long period of time and many solutions have been presented since. Path finding has many uses, from computer games to helping robots and rovers navigate through an environment [1]. There are many studies on robot-path planning using various approaches, such as the grid-based A* algorithm [2], road maps (Voronoi diagrams and visibility graphs) [3], [4], cell decomposition, and artificial potential field [5]. Some of the previous approaches [2],[6] use global methods to search the possible paths in the workspace, normally deal with static environments only, and are computationally expensive when the environment is complex[7].

The A* search algorithm used to find the shortest path between two points, was first introduced in 1968 and is still widely used today, especially in the interactive entertainment industry. The A* algorithm is guaranteed to find an optimal solution (assuming no negative costs and an admissible heuristic) [1].

In dynamic system, we introduced a simple yet efficient dynamic programming (DP) shortest path algorithm for real-time collision-free robot-path planning. The algorithm works in real time and requires no prior knowledge of target or barrier movements [7].

II. A* ALGORITHM [1]

The A* algorithm uses a heuristic to guide itself towards the goal. The heuristic estimates the cost to reach the goal node from the current node; the heuristic estimate is usually referred to as the h(n) value. The summary of the A* algorithm is illustrated in Figure 1.

A. Summary of the A Algorithm* [8]

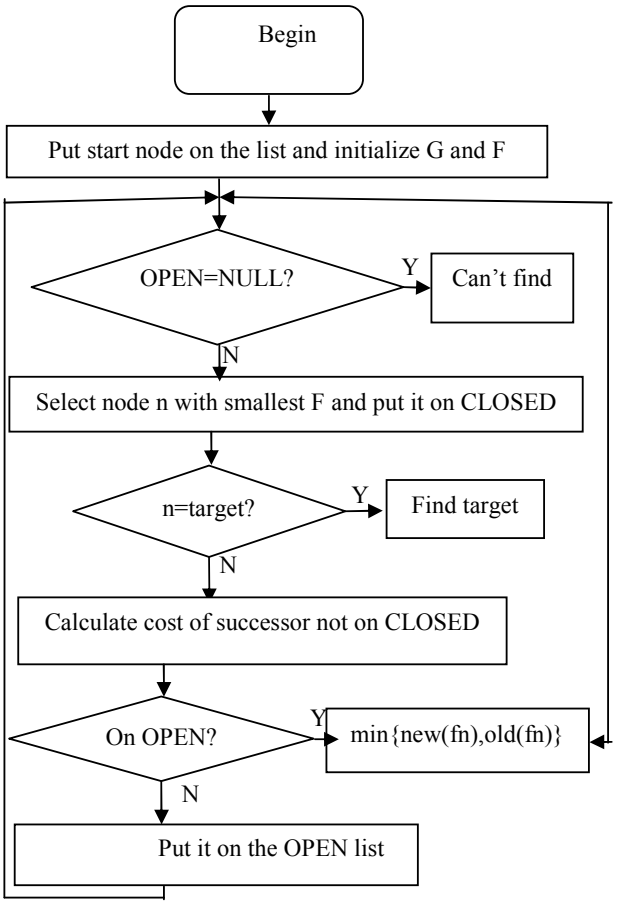


Figure 1. A* Algorithm

978-0-7695-4719-0/12 $26.00 © 2012 IEEE
DOI 10.1109/CSSS.2012.427

A* also keeps track of the cost needed to get to the current node from the start one, this cost is generally referred to as g(n). The total cost of a node, f(n), is the sum of the cost to reach the current node from the start node and the heuristic estimate.

B. *Heuristic* [1]

The heuristic part (the h(n) component) used by the A* algorithm is the most important part. If the heuristic function is admissible (meaning it never overestimates the minimum cost to the goal), then A* is also guaranteed to find the shortest/cheapest path. An ideal heuristic will always return the actual minimum cost possible to reach the goal. Three heuristics include Manhattan distance, Euclidean distance and Diagonal distance. The heuristic which is also commonly used is the Euclidean distance heuristic.

1) Manhattan distance heuristic(see Fig.2)

$$h_1(n)=|x_a - x_b| + |y_a - y_b|$$

The major drawback of the Manhattan heuristic is the fact that it tends to overestimate the actual minimum cost to the goal, which means that the path being found may not be an optimal solution.

2) Euclidean distance heuristic(see Fig.2)

$$h_2(n) = \sqrt{(x_a - x_b)^2 + (y_a - y_b)^2}$$

The Euclidean heuristic is admissible, but usually underestimates the actual cost by a significant amount. This means that we may visit too many nodes unnecessarily which in turn increases the time it takes to find the path.

3) Diagonal distance heuristic(see Fig.2)

$$h_3(n)=h_1(n)+(\sqrt{2}-2)*\min\{|x_a-x_b|,|y_a-y_b|\}$$

The diagonal distance heuristic combines aspects of both the Manhattan and Euclidean heuristics.

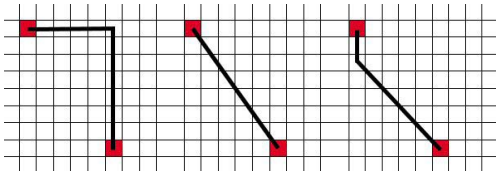


Figure 2. Three Heuristic Function

C. *Adjacencies* [1]

To find a path from the starting node to the goal node, we must define a way in which successor nodes can be selected. There are two common approaches: 4-adjacency (see Fig.3) and 8-adjacency (see Fig.3). In addition to 4 and 8 adjacency, we can also use 16-adjacency (see Fig.3) which allows even greater freedom of movement. Using 16-adjacency, we can also achieve a smoother looking road by reducing the sharpness of the turns.

Table 1 is the simulation and the results of the corresponding heuristic function using the 16-adjacency. Table 2 is the simulation with 8-adjacency.

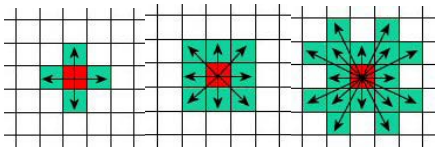


Figure 3. Three Adjacent Ways

TABLE I. SIMULATIONS OF THREE HEURISTIC FUNCTIONS

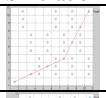
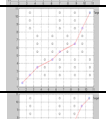
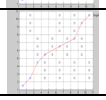
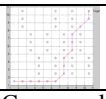
Simulation	Heuristic	visited nodes	Path Cost
	Manhattan	67	18
	Euclidean	49	13.1869
	Diagonal	52	13.8995

TABLE II. SIMULATION OF EUCLIDEAN HEURISTIC WITH 8-ADJACENCY

Simulation	Heuristic	visited nodes	Path Cost
	Euclidean	59	15.6569

Compared 8-adjacency with 16-adjacency of the result when using the Euclidean heuristic, we can find that the latter will be better (Other conditions are the same).

Since we are using 16-adjacency, the only admissible heuristic is the Euclidean distance, and it is the only one that never overestimates the minimum cost to get to the goal node and thus is guaranteed to find the shortest path.

III. DP ALGORITHM [7]

Like most robot-path-planning approaches, the environment is represented by a topologically organized map. Each grid point on the map has only local connections to its neighboring grid points from which it receives information in real time. The information stored at each point is a current estimate of the distance to the nearest target and the neighbor from which this distance was determined. Updating the distance estimate at each grid point is done using only the information gathered from the point's neighbors, that is, each point can be considered an independent processor, and the order in which grid points are updated is not determined based on global knowledge of the current distances at each point or the previous history of each point. The robot path is determined in real time completely from the information at the robot's current grid-point location. The computational effort to update each point is minimal, allowing for rapid propagation of the distance information outward along the grid from the target locations.

A. *Robot Environment*

Suppose the robot environment is discretized into a grid of *M* points, labeled by an index *i*, each point being either a free space or a barrier location, the targets and the robot may

occupy any free space. For example, consider a regular square grid as illustrated in Fig.4, which has a barrier at point six. The neighbor set for the point labeled seven in this figure and the distances from it to all other points are

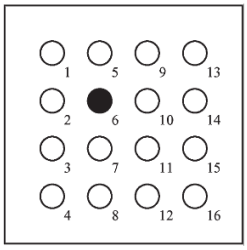


Figure 4. Regular square grid with barrier at location 6

$$B_7 = \{2,3,4,8,10,11,12\}$$

$$d_{7j} = \begin{cases} 1, & \text{if } j \in \{3,8,11\} \\ \sqrt{2}, & \text{if } j \in \{2,4,10,12\} \\ 2, & \text{if } j=15 \\ 1+\sqrt{2}, & \text{if } n \in \{1,9,14,16\} \\ 2\sqrt{2}, & \text{if } j \in \{5,13\} \end{cases}$$

From the above, we can know that this algorithm uses 8-adjacency to represent the robot environment.

B. Notations and Assumptions

In order to visualize the process, the following four assumptions have to be made.

- The robot and the target are replaced with the grid points.
- Update frequency of dynamic system is far greater than that of the target points and obstacles
- Robot's starting position is determined in advance and the location of obstacles and the target can be identified from the variables.
- Robot, the target point and the obstacles can move freely between the adjacent free points, but cannot change the direction of movement.

C. Algorithm

Planning in the presence of dynamic obstacles is still computationally challenging because it requires adding time as an additional dimension to the search-space explored by the planner. In order to avoid the increase in the dimensionality of the planning problem, most real-time approaches to path planning treat dynamic obstacles as static and constantly re-plan as dynamic obstacles move. This algorithm also uses the idea.

The general principle of distance propagation based dynamic path planning algorithm is that the distance information spreads out from the target point until it reaches the start point (location of the robot) in the grid environment. During the process each grid point records the minimum distance to target point and the robot moves along to the target point according to the information stored on each point.

D. The Principle of Extended DP Algorithm (EDP)

Extending the DP algorithm that builds on the observation that the distance between any two free spaces is defined to be the minimum Euclidean length of all paths joining the two points through nonbarrier adjacent neighbors. So, based on our previous the result of comparison, we can extend the DP algorithm with 16-adjacency. This can not only guarantee the optimality but also improve the search speed and expand the search area.

IV. SIMULATION

We set up a few scenarios to verify the feasibility and advantages of DP algorithm and EDP algorithm, both static and dynamic (either or both of the target and obstacles) environment will be discussed here. The simulations are done by MATLAB due to the limitation of conditions.

As shown in the following figures, obstacle points are marked with red solid dot, start point of robot are marked with red "+" and target point are marked with black solid dot.

For the simulation, other conditions are the same when running the DP algorithm and the EDP algorithm.

A. Static Environment

From the Fig.5 and the Fig.6, we can observe that the robot using the EDP algorithm obviously can reach the target point earlier no matter the number of steps or the length of the path.

B. Dynamic Environment

1) Moving obstacles and static target (see Fig.7 and Fig.8)

2) Both target and obstacles are moving (see Fig.9 and Fig.10)

From Fig.9 and Fig.10, We can see that the robot reaches the target before the moving obstacles move to the right to the border.

C. Performance analysis of the EDP algorithm

1) Time complexity

The EDP algorithm put forward in the paper only needs less computation, and we can imagine the speed of search from the above simulations. Search time of the EDP algorithm is only related to the size of the environment, so the EDP algorithm can be used in real-time.

2) Space complexity

The searching space of the EDP algorithm is only related with the size of the grid map. Grid information is stored through the two dimensional matrix, each grid describing its state wit rows, columns and assigned value. The number of

the grid addresses saved in the list of the path is decided by the length of the path, which is very limited and thus can be neglected. So the space complexity of EDP algorithm is O(n), n is the number of the grid points in the map.

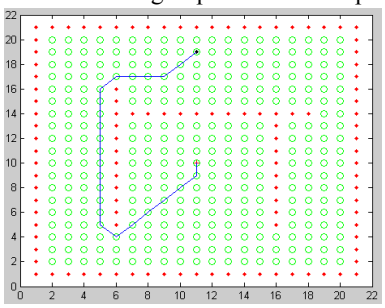


Figure 5. DP Algorithm

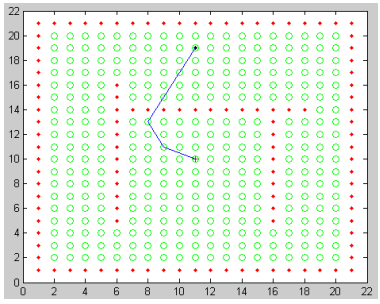


Figure 6. EDP Algorithm

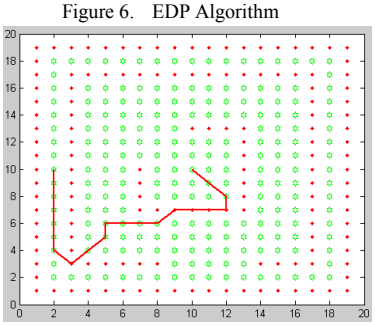


Figure 7. DP Algorithm

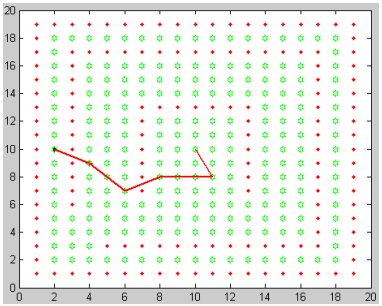


Figure 8. EDP Algorithm

V. CONCLUSION

A new EDP algorithm is proposed as an improvement of DP algorithm; and verified through several simulations. The local nature of the EDP algorithm distinguishes it from most algorithm, like A*, because it uses only local information [9], while A* algorithm uses global information. In this paper, we only extend the DP algorithm to 16-adjacency, thus it can expand the search area and improve the search speed, so the improved algorithm will be more suitable for more complex environment.

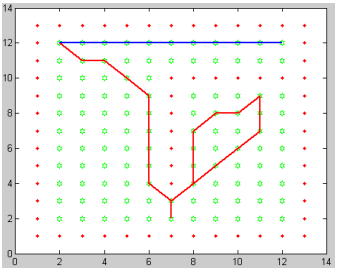


Figure 9. DP Algorithm

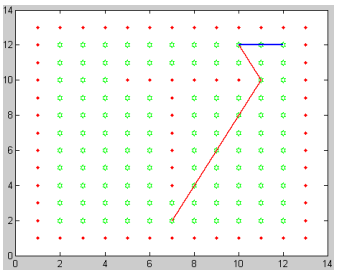


Figure 10. EDP Algorithm

REFERENCES

[1] D.R.Wichmann and B.Wuensche, Automated route finding on digital terrains [M], New Zealand, 2004.

[2] C. W. Warren, “Fast path planning using modified a∗ method,” in Proc. IEEE Int. Conf. Robotics and Automation, Atlanta, GA, May 1993,pp. 662–667..

[3] O. Takahashi and R. J. Schilling, “Motion planning in a plane using generalized voronoi diagrams,” IEEE Trans. Robot. Autom., vol. 5, no. 2, pp. 143–150, Apr. 1989..

[4] E. Hou and D. Zheng, “Mobile robot path planning based on hierarching hexagonal decomposition and artificial potential fields,” J. Robot. Syst.,vol. 11, no. 7, pp. 605–614, 1994..

[5] K.P.Valavanis, T.Hebert, R. Kolluru, and N. Tsourveloudis, “Mobile robot navigation in 2-d dynamic environments using an electrostatic potential field,” IEEE Trans. Syst., Man, Cybern. A, Syst., Humans, vol. 30, no. 2, pp. 187–196, Mar. 2000

[6] A. Zelinsky, “Using path transforms to guide the search for findpath in 2d,” Int. J. Rob. Res., vol. 13, no. 4, pp. 315–325, Aug. 1994

[7] Allan R. Willms and Simon X. Yang, Member, IEEE. An Efficient Dynamic System for Real-TimeRobot-Path Planning. IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS—PART B: CYBERNETICS, VOL. 36, NO. 4, AUGUST 2006. Pp: 755-766.

[8] Lester.P, “A* Pathfinding for Beginners”,2003
http://www.gamedev.net/page/resources/_/technical/artificial-intelligence/a-pathfinding-for-beginners-r2003.

[9] Allan R. Willms and Simon X. Yang, Member, IEEE. An Extension of the Distance-Propagating Dynamic System for Robot Path Planning to Safe Obstacle Clearance. Proceedings of the 2006 IEEE International Conference on Robotics and Biomimetics. December 17-20, 2006, Kunming, China. Pp: 1396-1401.