Path Planning in Service Robot Based on Improved A* Algorithm

Junyi Cao[a], Jie Liu[b]

College of Instrumentation and Electrical Engineering, Jilin University, Changchun 130000, China

[a]mr_caojunyi@163.com, [b]l_jie@jlu.edu.cn

Abstract. A* algorithm is a simple, scalable, and high reliable algorithm in the path planning algorithm. For the problem that service robots have high real-time characteristic and have short calculating time for algorithm, the A* algorithm is applied to the local path planning of service robots. To further improve the response speed of the algorithm, A* algorithm is improved by adjusting selection mechanism of the open update node choice and introducing the weight of Heuristics. The simulation and experimental results show that this new method has good planning performance. Adopting this new method can significantly reduce the searching time of algorithm and meet the high real-time need of service robots.

1. Introduction

In the food, transportation, electric power and other industries, service robots have a wide range of applications. Usually, the service robot[1] obtains the surrounding environment information and makes judgment according to the dynamic environment in which it is located, and avoids the obstacle timely in moving from the start position to the given target position. In the process of moving, the shortest planning path and the minimum time of consideration are required.

For path planning, the current mainstream approach is to abstract it into a graph search question [2]. Common searching methods are RRT algorithm, artificial potential field method, particle swarm optimization, A* algorithm and so on. As RRT algorithm [3], in the same environment multiple searching will lead to different paths, which is not conducive to the reuse of paths and the rapid re-planning, and even the searched paths will generally deviate from the optimal path. Artificial potential field method [4] is low in the ability dealing with complex environmental issue. It needs to be used in conjunction with other algorithms. Particle swarm method has disadvantages in local search ability, such as poor search and other shortcomings [5]. A* algorithm has the advantages of fast convergence speed, simple algorithm and high reliability, which can meet the requirements of service robots's stability, reliability and real-time [6]. However, in the place of particularly high real-time, A* algorithm has some limitations.

Through thorough study of the principles of A* algorithm, it can be improved in the open table node selection mechanism of algorithm updates, and introduce the weight of Heuristics. This new path planning algorithm is called WCAA* (Weight Convergence Accelerated A*) to further improve speed search in path planning. Finally the validity of improved algorithm in the paper is verified by Matlab software.

2. Improved A* Algorithm

The A* algorithm bases on the Dijkstra algorithm and uses heuristic information to guide the search. The algorithm has good performance, resulting profound impact in the field of heuristic search algorithms, and being widely used in related fields.

Firstly, there are two empty tables, which are open table and close table. Open table stores the node to be investigated and ready to search. Close table stores nodes which are already visited from the open table. The key of the algorithm's search is whether there is a minimum cost node in the open table when expand nodes.

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

In the mathematical formula: g (n) refers to the value from the start node n_{start} to the current node n. h (n) is the heuristic value from the current node n to the target node n_{target}. F (n) is the total value of the nodes from the start node n_{start} to the target node n_{target} via the current node n.

```
01. Main ()
02. OPEN=∅; CLOSE=∅;
03. g(n_start)=0;h(n_start)=Heuristics(n_start);f(n_start)=g(n_start)+h(n_start);
04. Insert n_start into OPEN with g (n_start) and f(n_start);
05. while OPEN≠∅ do
06.     remove n with the minimum f(n) from OPEN;
07.     if n≠n_target then
08.         CHILD=GenerateChildNodes (n);
09.         UpdateOpen (n, CHILD);
10.         insert n into CLOSE;
11.     else
12.         return SUCCESS;
13.         break;
14.     end if
15. end while
16. Update Open (n, CHILD)
17. for all n' ∈CHILD do
18.     if n' ∉CLOSE then
19.         c (n, n')=Cost(n,n');
20.         h (n')=Heuristics(n');
21.         if n'∈OPEN then
22.             if g(n')>g(n)+c(n,n') then
23.                 g (n')=g(n)+c(n,n');
24.                 f (n')=g(n')+h(n');
25.             end if
26.         else
27.             g (n')=g(n)+c(n,n');
28.             f (n')=g(n')+h(n');
29.             Insert n' into OPEN with g (n') and f (n');
30.         end if
31.     end if
32. end for
```

Figure 1. Code of A* algorithm

A* algorithm's code shows in Figure 1. The algorithm starts from the start node n_{start}, chooses the smallest node from the total value f (n) as the current node n, and then expands the node outwardly from the current node n until the target node n_{target}. When the target node becomes the current node, the path finding is successful. If the open table is empty, the path finding fails.

On the basis of the A* algorithm formula, set the weight in front of Heuristics (n'), namely h (N') =w*Heuristics (n') (20 row) to enhance the significance of the value of the heuristics. Compared with the A* algorithm, the path obtained by this way can't be guaranteed to be the shortest path. But it can converge to the target node quickly and shorten the calculating time.

A* algorithm updates the open table in this way: for each extended node n', calculate the historical value g (n') of the start node moving to the current node, h (n ') and the total value f (n '). If the node has been expanded before, the previous g (n ') is compared twice when it is expanded again, leaving the node with the smaller value of g (n '). The improved A* algorithm updating open table manner is different from the A * algorithm, namely f (n ')>g (n') + c (n, n') + h (n') (22 line). If the node has been expanded before, the previous f (n') is compared twice when it is expanded again, leaving the node with the smaller value of f (n'). In this way, A* algorithm could speed up the convergence, and improve search speed.

In the A* algorithm, node n enters the close table only once, and the close table is not updated anymore. However, since the heuristic value does not satisfy the consistency principle, there are two different implementation versions of the A* algorithm that sets up the weight of the heuristic value according to whether the node joins the close table and re-joins the open table. Pearl proves that even if the node is allowed to repeatedly enter the open table, setting the weight of the heuristic value of the A* algorithm returns the path quality does not exceed that comes from the A* algorithm itself [7].

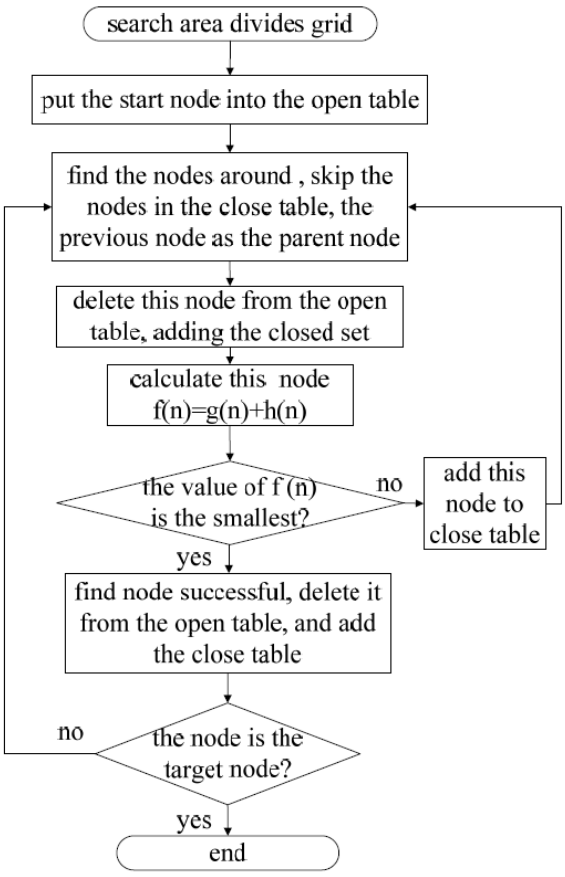

Figure 2. WCAA* algorithm flow chart

When the weight of the heuristic value is 0, WCAA* algorithm degenerates Dijkstra algorithm. When the weight w is between 0 and 1, the algorithm is weak heuristic. When the weight is 1, WCAA* algorithm degenerates to A* Algorithm. When the weight is greater than 1, the algorithm is to enhance the enlightenment. WCAA* algorithm flow chart shows in Figure 2.

3. Experimental Simulation and Analysis

The algorithm of this paper is implemented by Matlab software under Windows XP system. Software version is R2012a. Experimental hardware environment is Intel Core T4300 CPU, 2.10GHz, memory 2GB.

The service robot is 0.8m*0.3m in length and width and the average speed is 3.5m/s. The environment model is constructed by two-dimensional grid map. The map is 10.0m*10.0m and the resolution is 0.2m*0.2m.

Simulation results shows in Figure 3, Figure 4 and Figure 5.

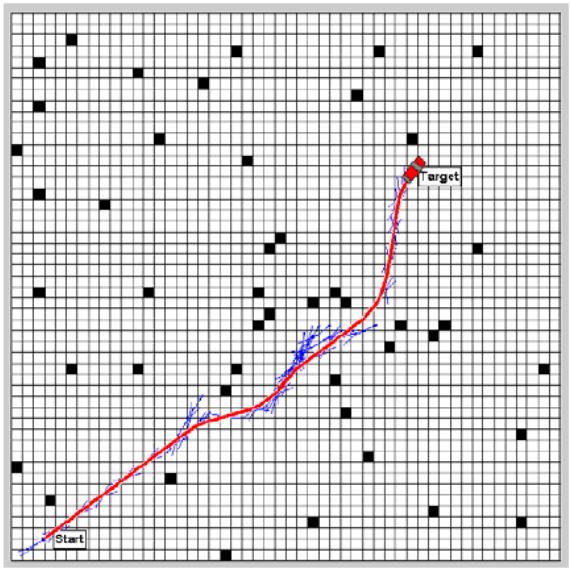

Figure 3. w=0.7

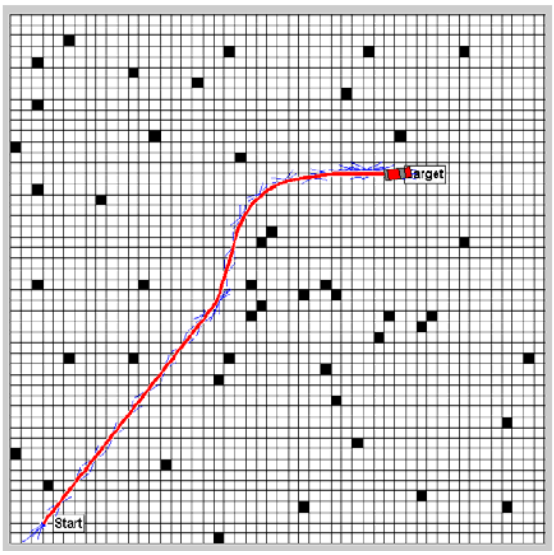

Figure 4. w=1.0

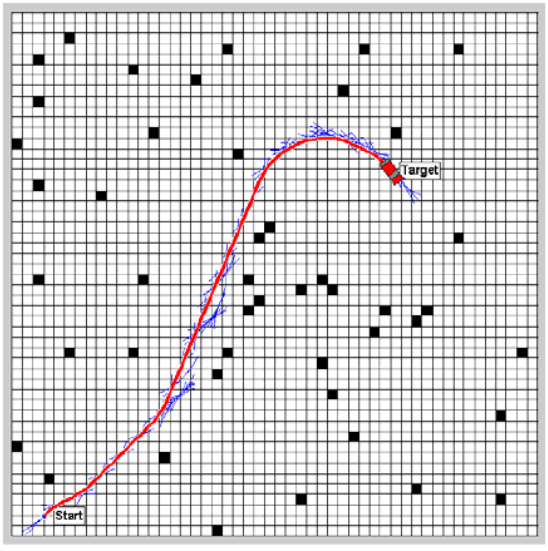

Figure 5. w=2.0

As shown in Table 1, when w = 1.0, the length of path from the start node to the target node is the shortest d = 5.08m and the search time is 0.355s. When w = 0.7, the length of path d = 5.25m and the search time is 0.273s decreasing by 23.1%. When w = 2.0, the planned path length d = 5.62m, search time 0.213s decreasing by 40%. Also, the number of expanding nodes is increasing, which results the increase of the path's length.

Table 1. Three experiment comparing

Heuristics	Start node	Target node	The number of Expanding nodes	The length of path (m)	Searching time(s)
0.7*H(n)	(3,2)	(36,35)	271	5.25	0.273
1.0*H(n)	(3,2)	(36,35)	242	5.08	0.355
2.0*H(n)	(3,2)	(36,35)	337	5.62	0.213

4. Conclusion

The real-time index plays an important role in the service robot's path planning. The improvement of real-time index needs to meet the high requirements of the service robot. This paper proposes a new algorithm which makes change in the way of updating the open table to speed up the convergence of the algorithm. Furthermore, by setting the weight of the heuristic value, the computation plan path time is drastically reduced. The simulation results show that the algorithm has achieved some improvements. The experimental data shows that the improved algorithm reduces the searching time by 40% compared with the basic A* algorithm.

Acknowledgments

This work was financially supported by fixed-wing time domain aviation electromagnetic system practical project (No.SS2013AA063903) fund.

References

[1] Doelling K, Shin J, Popa D O.Service robotics for the hone a state of the art review. Proceedings of Proceedings of the 7th International Conference on Pervasive Technologies Related to Assistive Environments. ACM, 2014, 35.
[2] Exact cell decomposition of arrangements used for path planning in robotics. Swiss Federal Institute of Technology, Institute of Theoretical Computer Science, 1999.
[3] La Valle S M. Rapidly-Exploring Random Trees A Цew Tool for Path Planning. 1998.
[4] Khatib O. Real-time obstacle avoidance for manipulators and mobile robots. The international journal of robotics research, 1986, 5 (1): 90 - 98.
[5] Eberhart R C, Kennedy J. A new optimizer using particle swarm theory[C]//Proceedings of the sixth international symposium on micro machine and human science. 1995, 1: 39 - 43.
[6] Hart P E, Nilsson N J, Raphael B. A formal basis for the heuristic determination of minimum cost paths. Systems Science and Cybernetics, IEEE Transactions on, 1968, 4(2): 100 - 107.
[7] Pear J. Heuristics: Intelligent search strategies for computer problem solving. 1984.