ISSN: 2664-9640

Path Planning for Indoor Mobile Robots based on Improved A * Algorithm

Hao Wu[a], Jie Liang[b]

School of Automation, Guangdong Polytechnic Normal University, Guangzhou 510665, China

[a]winter4131 @163.com, [b]1980136069@qq.com

Abstract

In different obstacle environments, the traditional A* algorithm may encounter the following issues during the path planning process: significant differences in efficiency among different heuristic functions, excessive number of search nodes, and non-smooth paths. This paper first combines the distance from a point to a line with the Euclidean distance, it is then integrated with a priority search strategy to improve search efficiency. Additionally, this paper proposes a turning point evaluation and node optimization algorithm to optimize the path. The result of experiments have shown that improved algorithm can stably reduce the number of traversed nodes, shortens the path length, and generates smoother planned paths, enabling the mobile robot to reach the destination efficiently and safely.

Keywords

A* Algorithm; Path Optimization; Path Planning; Mobile Robots.

1. Introduction

Navigation, as one of the important functions of mobile robots, has been extensively studied by researchers. Mobile robot navigation can be divided into four major steps: environmental perception, self-localization, path planning, and motion control. Path planning can be broadly divided into global path planning and local path planning. Global path planning focuses on obstacles in a static environment and involves various algorithms such as A* algorithm [1, 2], Dijkstra algorithm [3], Ant Colony Optimization(ACO) [4], Genetic Algorithm (GA) [5], and more. The A* algorithm has become one of the classic heuristic search algorithms due to its high planning efficiency. In A* algorithm, the robot's direction is artificially limited to fixed angles, resulting in paths with excessive turning points and redundant nodes. Researchers have proposed path planning algorithms that are not restricted by angles to overcome limitations on robot turning angles, including the theta* algorithm [6], Rapidly-exploring Random Tree (RRT) algorithm [7], and D* Lite algorithm [8]. To improve the search efficiency of A* algorithm, in paper [9, 10], they made improvements to the heuristic function.

In order to reduce the number of turning points and improve the smoothness of the path, other algorithms or mechanisms can be introduced into path planning to improve path smoothness [11]. For example, by incorporating an angle determination mechanism to effectively remove redundant nodes in the path [12]. Additionally, integrating collision cones methods [13], as well as combining B splines and Bezier curves methods [14, 15]. Although these algorithms can effectively address the path planning requirements of mobile robots, when dealing with different environments, they exhibit issues such as slow convergence speed, a high number of search nodes, excessive turning points, and non-smooth paths.

To sum up, we propose the improved algorithm to address the shortcomings mentioned in the literature. We propose an algorithm based on priority search strategy. By combining the distance from a point to a line and the Euclidean distance as heuristics, and we propose a

ISSN: 2664-9640

turning point evaluation function. By evaluating the turning points in the path and considering them as key points, we regenerate the path, thereby reducing the number of turning points in the path.

2. Path Planning based on A* Algorithm

2.1. Heuristic Function

The cost function of the improved A* algorithm consists of the actual cost function and the heuristic function, the cost function of the A* algorithm is described as:

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

Where g(n) is the actual cost from the starting point to the current node, and h(n) is the estimated cost from the current node to the goal.

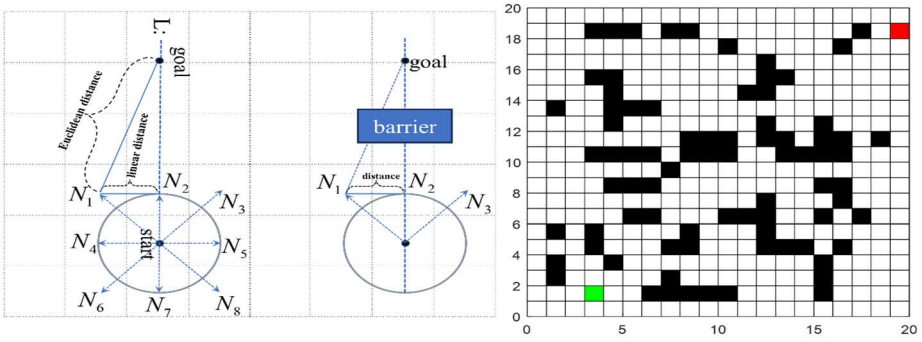
The selection of a heuristic function determines the search efficiency of the algorithm. When $h(n) > g(n)$, the number of traversed nodes is less, but the planned path may be suboptimal. When $h(n) < g(n)$, the algorithm can find the optimal path, but it may search through a larger number of nodes. When $h(n) = g(n)$, the search efficiency is highest and the results are best. The presence of obstacles leads to an imbalance between $h(n)$ and $g(n)$. In paper [9], they proposed a heuristic function that combines point to line distance with Euclidean distance, as shown in the following formula.

$$\begin{cases} h(n) = d_1 * h_1(n) + d_2 * h_2(n) \\ h_1(n) = \left| x_n * a + y_n * b + c \right| / sqrt(a * a + b * b) \\ h_2(n) = sqrt((x_n - x_N)^2 + (y_n - y_N)^2) \\ L: a * x + b * y + c = 0 \end{cases} \tag{2}$$

Where $h_1(n)$ represents the distance from point $P_n(x_n, y_n)$ to the line L, (a,b,c) is the parameter expression of the line L formed by the starting point and the ending point. $h_2(n)$ is the Euclidean distance from point P_n to the ending point $P_n(x_N, y_N)$. d_1 and d_2 are coefficients.

2.2. Priority search strategy

In a grid map, the A* algorithm typically sets the search direction to either 8 or 4 directions. During the path search, calculating the cost in a fixed direction each time leads to an issue of excessive search nodes. To improve search efficiency, a priority search strategy can be implemented. By prioritizing the expansion of the search from the start point towards the goal.



(a)Search direction of node (b) Grid map

Figure 1. Node search process

ISSN: 2664-9640

During the path search, calculating the cost in a fixed direction each time leads to an issue of excessive search nodes. To improve search efficiency, a priority search strategy can be implemented.

Table 1. Calculation of cost for neighborhood nodes

Coordinates	$F(n)$	$G(n)$	$H(n)$	Increment
$N_1(3,3)$	23.9606	1.4142	22.5163	0.5786
$N_2(4,3)$	23.2255	1.0000	22.2255	-0.1265
$N_3(5,3)$	23.3710	1.4142	21.9586	0.0190
$N_4(3,2)$	24.6146	1.0000	23.6146	1.2626
$N_5(5,2)$	24.0986	1.4142	23.0986	0.7466
$N_6(3,1)$	26.1479	1.0000	24.7337	2.7959
$N_7(4,1)$	25.4851	1.4142	24.4851	2.1331
$N_8(5,1)$	25.6739	1.0000	24.2596	2.3219
$N_c(4,2)$	23.3520	0.0000	23.3520	0.0000

The direction of goal can be inferred from the cost of nodes. As shown in Figure 1, the goal point is located at the upper right of the start point. The green node represents the start point, and the red node represents the goal point. According to the formula (1) and (2), the cost of eight neighboring nodes of the start point can be calculated, as shown in the Table 1. By subtracting the cost of current node N_c from the cost of the neighboring nodes, the cost increment for that direction can be obtained. By limiting the cost increment of the nodes, the algorithm can prioritize traversing nodes with lower cost. When the increment of the nodes is limited to 0.5, N_2 and N_3 will be searched firstly. Only when the search along the N_2 and N_3 directions is interrupted, will the other directions of the nodes be searched.

As shown in Figure 1(a), each node represents a direction. The values in each direction represent the cost of that direction. If the target is located above the node, the priority expand directions are upper, upper left and upper right. If the search is interrupted during this process, other directions will be searched.

The search is prioritized towards directions with smaller costs of child nodes. It does not aim to improve search speed by reducing search directions. Instead, during the traversal of child nodes, the ones with higher costs are marked and placed in the CloseList. When the search is interrupted, some marked nodes with smaller costs are taken from the CloseList to resume the search. The following is a flowchart for the priority search algorithm,as shown in Figure 2.

ISSN: 2664-9640

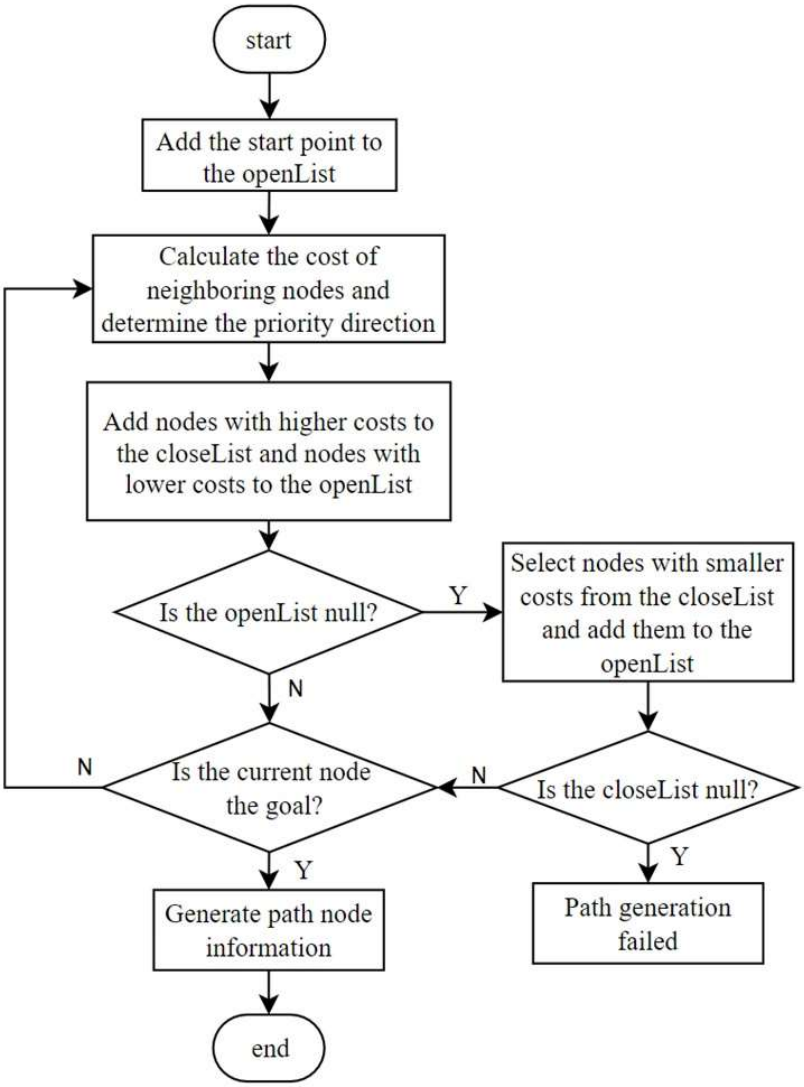


Figure 2. The flowchart of priority search strategy

Commonly used heuristic functions include Euclidean distance, Manhattan distance, Diagnoal distance and Chebyshev distance. The efficiency of the search varies greatly depending on the specific characteristics of obstacle maps. In Figure 3, which shows the search performance between different heuristic functions. Considering the actual size of the robot, when conducting global path planning, it is necessary to plan a path that is far from obstacles to reduce the chance of collision during local path planning. Therefore, the obstacles in the map need to be inflated. The green area represents blank space, the purple area represents the inflated region, the black area represents obstacles, and the white area represents unknown regions. The light blue area represents the nodes traversed by the calculation once, and the dark blue area represents the nodes traversed many times.

From the Figure 3, it can be seen that by implementing a priority search strategy, the search efficiency has been improved by approximately -9%, 67.3%, 3.9% respectively, compared to the case of using the combination of the distance from a point to a line and the Euclidean distance. In different maps, the number of search nodes is more stable and fewer than other heuristic distance, resulting in an overall improvement of 62.2% in search efficiency.

ISSN: 2664-9640

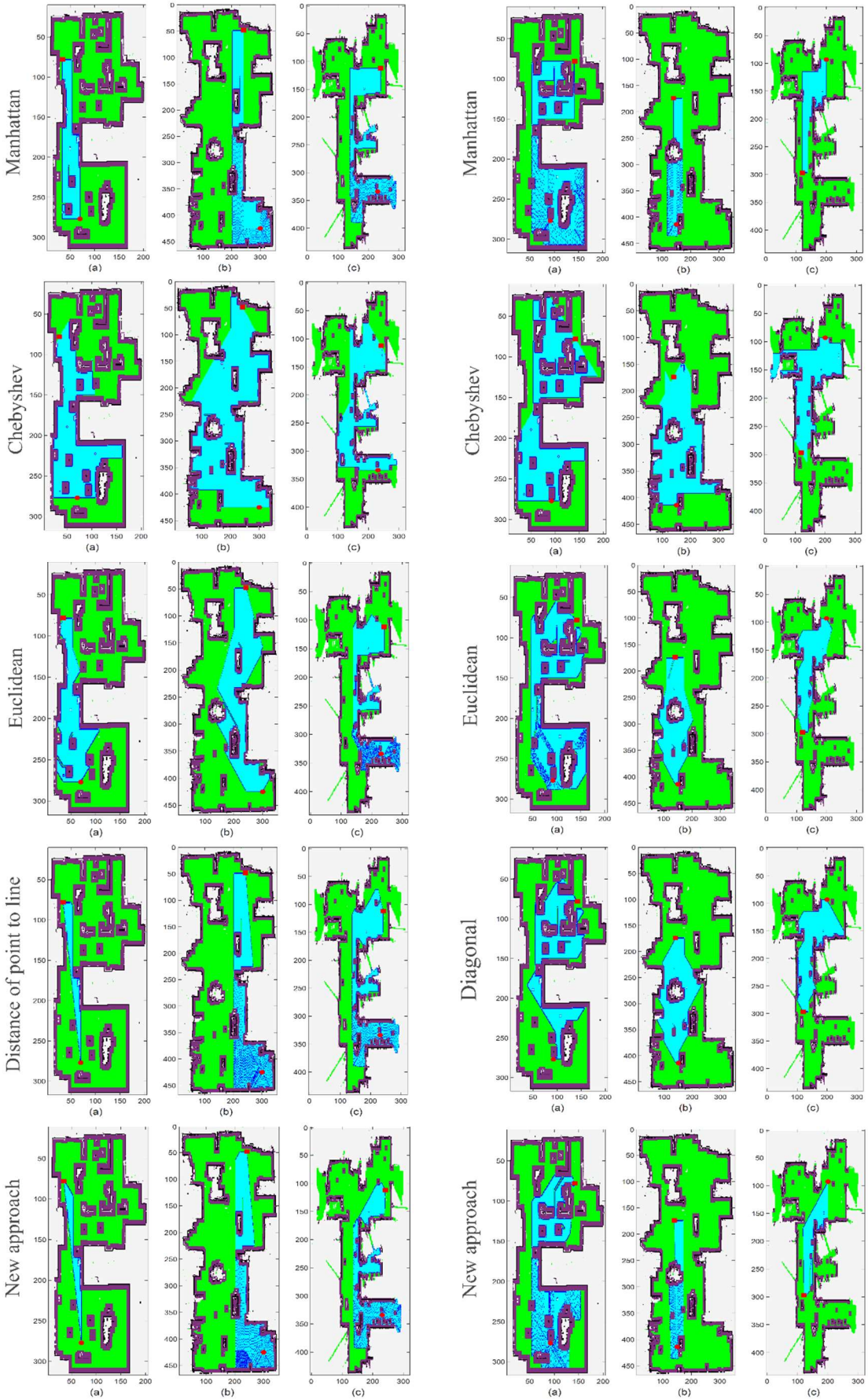


Figure 3. Performance of different heuristic functions

ISSN: 2664-9640

3. Path Planning based on A* Algorithm

Due to the restrictions imposed by fixed turning angles, the grid path in the planned path is suboptimal and contains unnecessary nodes. To tackle this problem, this section presents a novel algorithm for optimizing the path. The path generated based on global path planning is represented by $path = \sum_1^n (P_i)$, where P_1 to P_n are the nodes of the path information.

In the grid map , The continuity of nodes can be determined using the following formula.

$$\begin{cases} \overrightarrow{P_2P_3} = P_3(x_3, y_3) - P_2(x_2, y_2) \\ \overrightarrow{P_1P_2} = P_2(x_3, y_3) - P_1(x_2, y_2) \\ \overrightarrow{P_1P_3} = P_3(x_3, y_3) - P_1(x_1, y_1) \end{cases} \tag{3}$$

If formula $\overrightarrow{P_2P_3} = \overrightarrow{P_1P_2}$ && $2\overrightarrow{P_1P_2} = \overrightarrow{P_1P_2}$ holds, it is considered a continuous point; otherwise, it is considered a turning point. By repeating this process, we can identify all the turning points along the path and designate them as key points. This allows us to collect all the turning points in the path, as illustrated in Figure 4.

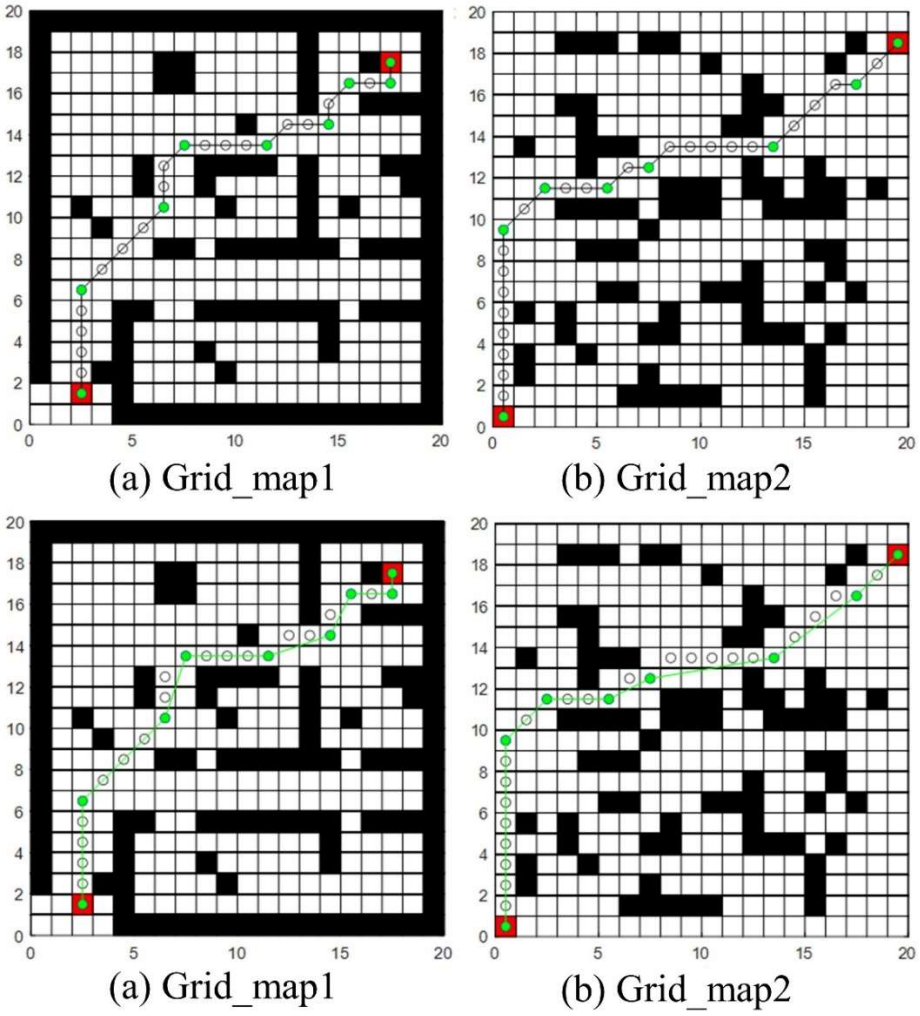


Figure 4. Turning point evaluation and path optimization

ISSN: 2664-9640

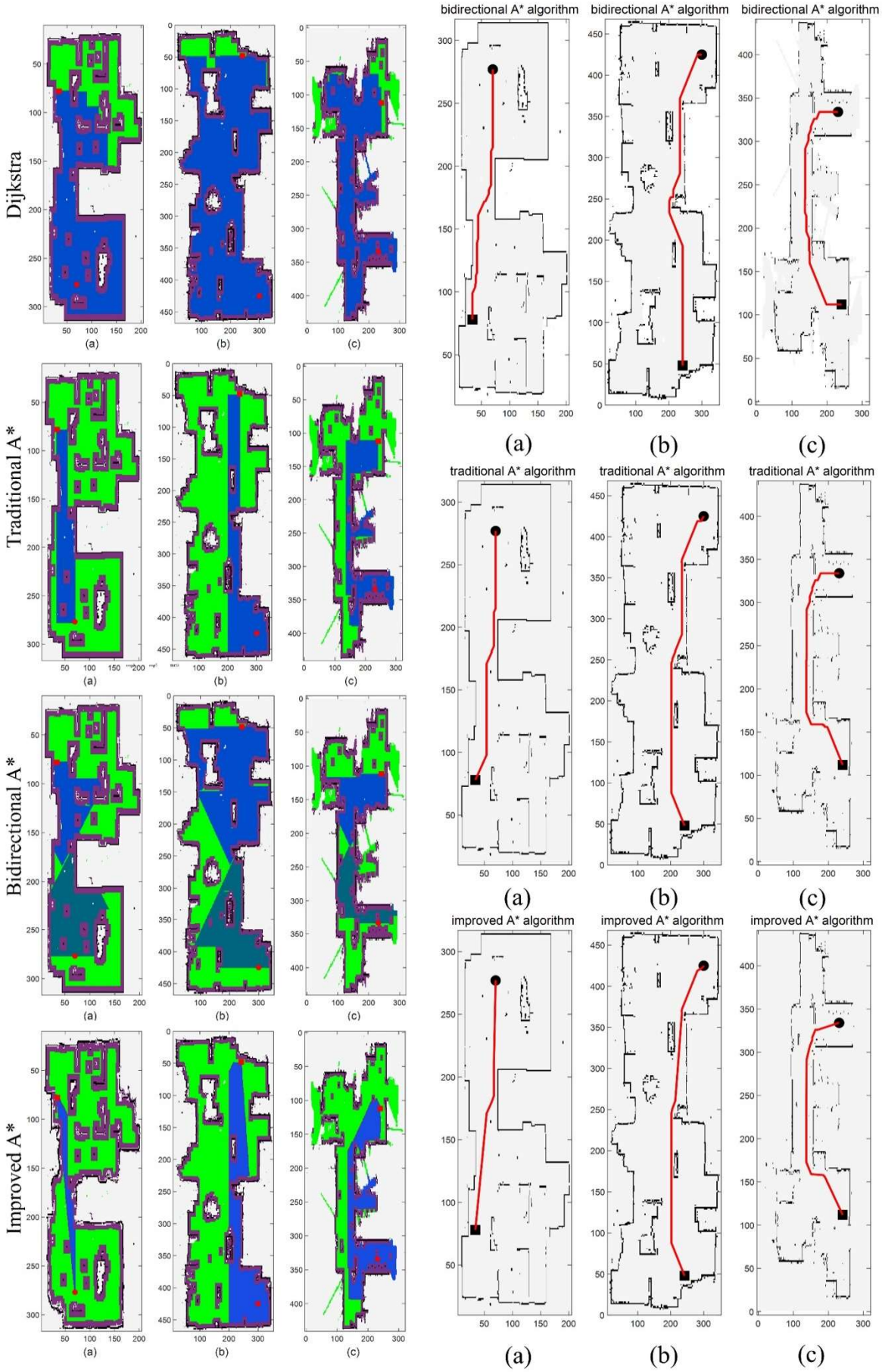


Figure 5. Performance under different algorithms

ISSN: 2664-9640

After obtaining all the turning points in the path, it is necessary to determine whether optimization is possible between different turning points. Sample points should be taken based on the distance between the turning points, and these sample points should be checked to see if they pass through or are close to obstacles. If these points are not close to obstacles, the path should be optimized and regenerate a new path. Otherwise, the original path information should be retained.

4. Experimental Comparison and Analysis

To validate the algorithm's superiority, we performed experimental comparisons using traditional A* algorithm, Dijkstra algorithm, and bidirectional A* algorithm on three distinct maps. The number of search nodes was statistically analyzed and presented in Figure 5. Both the traditional A* algorithm and bidirectional A* algorithm employed the Manhattan distance as their heuristic function.

Table 2. The specific performance of different algorithms

Algorithm	Nodes	Length of path	Time(s)	Turning
Dijkstra A*	37618	306	0.885966	18
Bidirection A*	23902	308	0.521772	23
Traditional A*	12108	306	0.227782	18
Improved A*	10124	282	0.214671	14

We separately recorded the number of search nodes, path length, time, and the number of turnings. The following data represents the average values across three maps.The comparison between path length and the number of turning points under different algorithms is shown in Figure \ref{fig6}. The specific performance comparison of different algorithms is shown in Table 2. Analysis of the data reveals that the Dijkstra algorithm, traditional A* algorithm, and improved A* algorithm yield superior global paths.

Analysis of the data reveals that the Dijkstra algorithm, traditional A* algorithm, and improved A* algorithm yield superior global paths. However, the Dijkstra algorithm, as a traversal algorithm, requires a large number of search nodes and thus requires more time. In contrast to the traditional A* algorithm, the bidirectional A* algorithm demonstrates an increased number of search nodes, longer paths, and more turning points. Conversely, the improved A* algorithm decreases the number of search nodes by 16.4% and achieves an average path length reduction of 7.8%. It also reduces search time by 5.8% and decreases the number of turning points in the path by 28.6%.

5. Conclusion

This paper proposes a new indoor mobile robot path planning method. Based on the A* algorithm, this method effectively reduces the number of traversed nodes, decreases the number of turning points in the path, and makes the path smoother. Moreover, for scenarios where obstacles obstruct the global path, it ensures that the robot can safely reach the target and avoids getting stuck in local optimal solutions. Experimental and comparative results validate the effectiveness of the proposed algorithm. Research direction in the future include multi-robot collaboration, which can greatly improve the efficiency of robots. By utilizing high-precision sensors and sensor fusion techniques, the navigation and localization of robots can be more accurate, enabling them to timely avoid obstacles. However, as robots encounter increasingly complex environments, the importance of three-dimensional path planning is

ISSN: 2664-9640

being emphasized. Research in this field is becoming more comprehensive and sophisticated, with the goal of meeting a wide range of needs.

References

[1] P. E. Hart, N. J. Nilsson and B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," in IEEE Transactions on Systems Science and Cybernetics, vol. 4, no. 2, pp. 100-107, July 1968.

[2] Zhong, Xunyu, et al. "Hybrid path planning based on safe A* algorithm and adaptive window approach for mobile robot in large-scale dynamic environment." Journal of Intelligent & Robotic Systems 99.1 (2020): 65-77.

[3] Đakulović, Marija, Mijo Čikeš, and Ivan Petrović. "Efficient interpolated path planning of mobile robots based on occupancy grid maps." IFAC Proceedings Volumes 45.22 (2012): 349-354.

[4] Lyridis, Dimitrios V. "An improved ant colony optimization algorithm for unmanned surface vehicle local path planning with multi-modality constraints." Ocean Engineering 241 (2021): 109890.

[5] Song, Baoye, Zidong Wang, and Lei Zou. "On global smooth path planning for mobile robots using a novel multimodal delayed PSO algorithm." Cognitive Computation 9.1 (2017): 5-17.

[6] Han, Sen, et al. "A dynamically hybrid path planning for unmanned surface vehicles based on non-uniform Theta* and improved dynamic windows approach." Ocean Engineering 257 (2022): 111655.

[7] Qian, Kui, et al. "Robot path planning optimization method based on heuristic multi-directional rapidly-exploring tree." Computers & Electrical Engineering 85 (2020): 106688.

[8] Aizat, Muhammad, Ahmad Azmin, and Wan Rahiman. "A survey on navigation approaches for automated guided vehicle robots in dynamic surrounding." IEEE Access 11 (2023): 33934-33955.

[9] Liu, Haoxin, and Yonghui Zhang. "ASL-DWA: An improved A-star algorithm for indoor cleaning robots." IEEE Access 10 (2022): 99498-99515.

[10] Ju, Chunyu, Qinghua Luo, and Xiaozhen Yan. "Path planning using an improved a-star algorithm." 2020 11th International Conference on Prognostics and System Health Management (PHM-2020 Jinan). IEEE, 2020.

[11] Ali, Hub, et al. "Path planning of mobile robot with improved ant colony algorithm and MDP to produce smooth trajectory in grid-based environment." Frontiers in neurorobotics 14 (2020): 44.

[12] Ma, Guojun, et al. "A probability smoothing Bi-RRT path planning algorithm for indoor robot." Future Generation Computer Systems 143 (2023): 349-360.

[13] Kim, Yong Hwi, et al. "Smooth path planning by fusion of artificial potential field method and collision cone approach." MATEC Web of Conferences. Vol. 75. EDP Sciences, 2016.

[14] Huang, Hsu-Chih. "FPGA-based parallel metaheuristic PSO algorithm and its application to global path planning for autonomous robot navigation." Journal of Intelligent & Robotic Systems 76 (2014): 475-488.

[15] Song, Rui, Yuanchang Liu, and Richard Bucknall. "Smoothed A* algorithm for practical unmanned surface vehicle path planning." Applied Ocean Research 83 (2019): 9-20.