Journal of Artificial Intelligence Practice (2024)
Clausius Scientific Press, Canada

DOI: 10.23977/jaip.2024.070307
ISSN 2371-8412 Vol. 7 Num. 3

Research on path planning of patrol robot based on multi-algorithm fusion

Xiancheng Fan[1,*], Yeqing Yu[1], Tao Li[1]

[1]*School of Electrical and Electronic Engineering, Anhui Institute of Information Technology, Wuhu, China*

[*]*Corresponding author: 2382493123@qq.com*

Keywords: Patrol robots, Improved A*, Floyd algorithm, Improved DWA, Path planning

Abstract: A multi-algorithm fusion path planning algorithm for patrol robots was proposed, In order to improve the robot's path planning ability, optimize the search efficiency, improve the robot's path smoothness and improve the control accuracy. The A* algorithm is optimized through the search field and heuristic function to optimize the node search, avoid the expansion of redundant nodes and improve the search efficiency of the algorithm while ensuring the optimal global path. The improved A* algorithm still has node redundancy, excessive path transition and other phenomena. Floyd algorithm is used to introduce improved A* key nodes to optimize the improved A* algorithm again, eliminate redundant nodes, smooth the global path, and dynamically increase the number of key nodes for long-distance key nodes to effectively prevent path deviation. In view of the shortcomings of the improved A* algorithm in dynamic obstacle planning, the improved DWA algorithm is integrated to achieve local path planning, and the integrated path planning algorithm has local dynamic and unknown environment obstacle avoidance ability. Experiments show that the proposed fusion algorithm has the ability of global path planning and local path planning, which verifies the feasibility and effectiveness of the fusion algorithm.

1. Introduction

Patrol robots play an increasingly important role in modern society, they are widely used in security, monitoring, rescue and other fields. As one of the core functions of patrol robot, path planning directly affects its performance and efficiency. Path planning of patrol robots is a very complicated and practical problem, which involves from finding the optimal path in static map to adjusting the path in dynamic environment to cope with various complex external environment. The traditional path planning algorithm usually only considers the optimal path in static environment, but ignores the influence of dynamic environment on path planning, which is difficult to adapt to the real-time changes of the environment, and easily leads to the inconsistency between the planned path and the actual situation, which affects the efficiency and performance of patrol robots.

With the deepening of the research on robot path planning, path planning is divided into global path planning and local path planning. Global path planning belongs to static planning and local path planning belongs to dynamic planning. Among them, the common algorithms for global path planning include Dijkstra algorithm[1], A* algorithm[2], D* algorithm[3] and other algorithms

based on graph search, RRT algorithm based on sampling[4], and intelligent algorithms such as ant colony algorithm[5], genetic algorithm[6], neural network algorithm[7] and so on. Local path planning algorithms commonly used artificial potential field method[8], dynamic window algorithm (DWA)[9], time elastic band algorithm (TEB)[10] and other algorithms. Global path planning is the optimal path planning for static maps, which cannot meet the path planning in complex dynamic environments. However, the method based on local planning may lead to local optimal path rather than global optimal path, thus limiting the navigation ability of robots [11]. In view of this situation, a single algorithm has been unable to meet people's needs for complex environment applications, and the fusion algorithm of global path planning and local path planning has become the hot core of robot path planning research. Among them, the fusion algorithm of A* and artificial potential field can realize the robot's global path and local path planning, but fails to take into account the influence of dynamic obstacles on the algorithm [12].The fusion algorithm of A* and DWA can realize dynamic local programming, but A* algorithm still has redundant nodes that need to be further optimized [13]. The fusion of A* and intelligent algorithm has high complexity, long planning time and low efficiency [14].

Therefore, this paper proposes A multi-algorithm fusion path planning algorithm for patrol robots, which optimizes A* algorithm through search domain and heuristic function, and optimizes node search to avoid the expansion of redundant nodes and improve the search efficiency of the algorithm under the condition of ensuring the optimal global path. The improved A* algorithm still has node redundancy, excessive path transition and other phenomena. Floyd algorithm is used to introduce improved A* key nodes to optimize the improved A* algorithm again, eliminate redundant nodes, smooth the global path, and dynamically increase the number of key nodes for long-distance key nodes to effectively prevent path deviation. In view of the shortcomings of the improved A* algorithm in dynamic obstacle planning, the improved DWA algorithm is integrated to achieve local path planning. The integrated path planning algorithm has the ability to avoid obstacles in local dynamic and unknown environments, and completes local path planning while ensuring the global optimization.

2. Classical A* algorithm research

2.1 Classical A* algorithm

A* algorithm can be traced back to the 1968 paper "A Formal Basis for the Heuristic Determination of Minimum Cost Paths" was first proposed, A heuristic search algorithm, Used to find the shortest path from the starting point to the target point in a graph or graph class problem [15]. A* algorithm combines breadth-first search (BFS) with Dijkstra's algorithm and introduces a heuristic search algorithm to measure the distance relationship between the real-time search location and the target location, so that the search direction is preferentially oriented to the direction of the target point, and finally achieves the effect of improving search efficiency [16].

The classic A* algorithm is divided into the following steps:

Setp 1: System initialization: create two lists open list and close list. The open list stores the nodes to be checked, and the close list stores the nodes that have been searched. Select the start node, set the cost of the start node to zero, and put it in the open list. At the same time, the generation value to the target node is estimated and recorded by a heuristic function according to each node position.

Setp 2: Node selection: Initially, the start node is the only candidate waiting node.

Setp 3: Node extension: For the selected node, traverse its domain nodes. Calculate the actual cost of getting from the current node to the domain node and add the estimated cost of getting from the domain node to the target node. If this total cost is less than the current total cost of the domain

node, update the total cost of the domain node and set the current node as the parent of the domain node.

Setp 4: Node tag: Places the selected node tag in the closed list and removes it from the open list.

Setp 5: Search loop: Repeat steps 2 to 4 until the target node is found or the open list is empty. The loop ends. If the target point appears in the open list, the path is found; if the open list is empty, the path is not found.

Setp 6: Path backtracking: Once the destination node is found, the shortest path can be built from the destination node back to the start node.

The advantage of depth-first search is that it is fast, but it can rarely find the optimal solution. Breadth-first search can indeed find the optimal solution, but because breadth-first search is layer by layer, covering every node, the search time is long and the space efficiency is not high. A* algorithm can extract the advantages of both and solve these two disadvantages: not only to find the best solution with great probability, but also to reduce the redundant time, the most critical part of the advantage of A* algorithm is the determination of its heuristic function.

2.2 Search domain simulation comparison

The classical A* search domain, 8-neighborhood search method is adopted. As shown in Figure 1, node S is the starting point of the robot, and the robot can move in 8 directions from node 1 to node 8 at S. Now the search direction is optimized according to the relationship between the connection between the robot's current position and the target point and the Angle formed by the direction of node 2, so as to reduce unnecessary search nodes and improve search efficiency. The node search method has been optimized from the original 8-neighborhood search to the variable neighborhood search method [17]. In the variable neighborhood search method, the smallest neighborhood search is first adopted, and when the solution cannot be improved, the solution is switched to a slightly larger neighborhood. If the solution can continue to be improved, the solution is returned to the smallest neighborhood; otherwise, the search is continued to be switched to a larger neighborhood, and the search diversity is increased by changing the neighborhood structure, thus increasing the optimization probability [18], through the A* Manhattan distance search method, the search methods of 4-neighborhood, 8-neighborhood and text change neighborhood are simulated. As shown in Figure 2, where black is the obstacle, white is the free area, gray is the traversal node, red line is the planned path, blue dot represents the starting point, red dot represents the end point. Figure 2-a shows 4-neighborhood search path planning, 2-b shows 8-neighborhood search path planning, and 2-c shows variable neighborhood search path planning. Compare the neighborhood algorithms, as shown in Table 1.

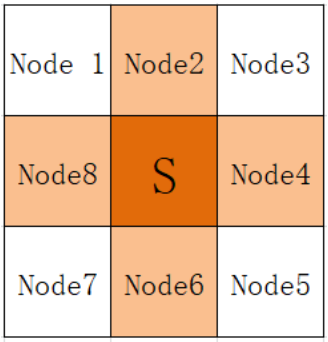
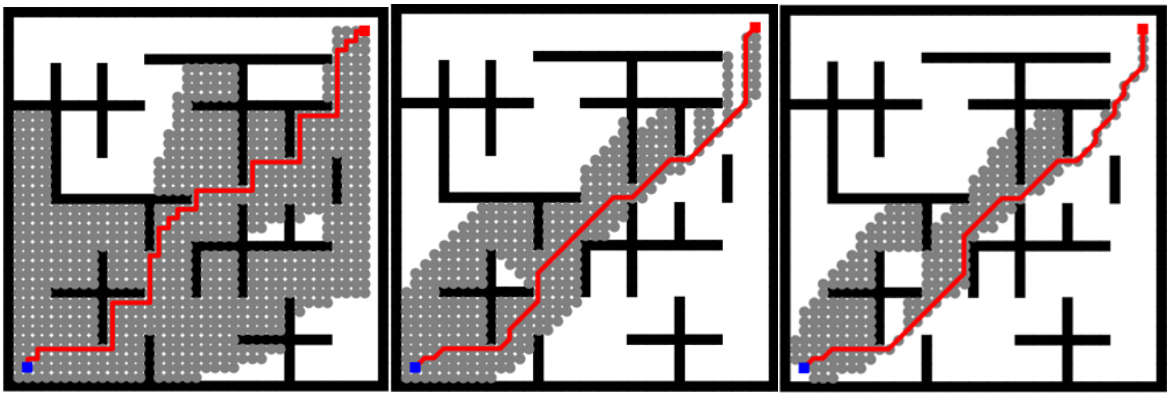


Figure 1: Neighborhood graph



(a) (b) (c)

Figure 2: Neighborhood simulation diagram

Table 1: Comparison table of neighborhood search algorithms

Algorithm	Simulation time (s)	Number of traversal nodes	Planning distance
4 Neighborhood	2.980	855	72.0
8 Neighborhood	1.613	373	57.355
Variable neighborhood	1.342	259	57.355

According to Figure 2 and Table 1, compared with the four-neighborhood search algorithm, the variable neighborhood search strategy adopted in this paper saves 54.97% in simulation time, 69.71% in search nodes and 14.64% in planned paths; compared with the 8-neighborhood search algorithm, it saves 16.80% in simulation time and 30.56% in search nodes. All planned paths can be planned to the optimal path. The variable neighborhood search strategy selected in this paper can find the optimal path, and is obviously superior to the 4-neighborhood search strategy and the 8-neighborhood search strategy in simulation time and search node number.

2.3 Heuristic function simulation comparison

The traditional A* algorithm evaluation function is:

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

In formula (1), g (n) represents the actual generation value from the starting point to the current node, and h(n) represents the estimated generation value from the current node to the target point. Commonly used predictive cost functions include Manhattan distance, Euclidean distance and Chebyshev distance[18], while the standard heuristic function for A* is Manhattan distance, and the Manhattan distance function is:

$$h_M = \left|x_c - x_g\right| + \left|y_c - y_g\right| \tag{2}$$

In formula (2), x_c and y_c is the horizontal and vertical position coordinates of the current node, x_g and y_g is the horizontal and vertical position coordinates of the target node.

In A complex environment, the path chosen by the A* algorithm is not optimal, and the A* algorithm is faced with the shortcomings of long search time, low search efficiency, and many redundant nodes. In the face of this situation, the heuristic function of A* algorithm is improved, and Euclidean distance and Chebyshev distance are proposed to replace Manhattan distance

function [19].

The Euclidean distance heuristic function is:

$$h_E = \sqrt{\left(x_c - x_g\right)^2 + \left(y_c - y_g\right)^2} \tag{3}$$

The Chebyshev distance heuristic function is:

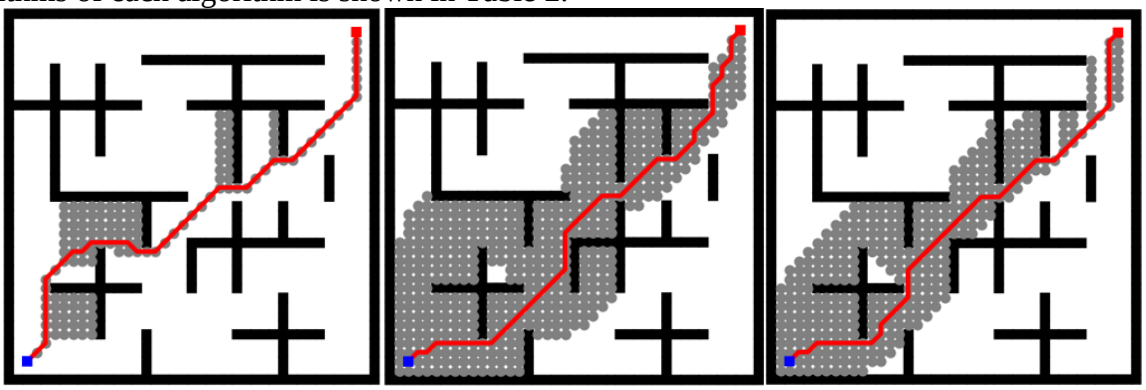
$$dx = \left|x_c - x_g\right| \tag{4}$$

$$dy = \left|y_c - y_g\right| \tag{5}$$

$$h_C = dx + dy + \left(\sqrt{2} - 2\right) * \min(dx, dy) \tag{6}$$

Where dx and dy are the difference between the current coordinate and the target point.

Manhattan distance function, Euclidean distance and Chebyshev distance were simulated and compared respectively, as shown in Figure 3-a, 3-b and 3-c. The comparison of the path planning algorithms of each algorithm is shown in Table 2.



(a) (b) (c)

Figure 3: Algorithm simulation diagram

Table 2: Algorithm Comparison Table

Algorithm	Simulation time (s)	Number of traversal nodes	Planning distance
A*- Manhattan	0.981	142	59.541
A* - Euclidean	2.578	512	57.355
A* - Chebyshev	1.632	373	57.355

As can be seen from Figure 3 and Table 2, the A* -Manhattan distance algorithm has high path planning efficiency and few search nodes, but it cannot find the optimal path. A*-Euclidean distance algorithm, the planning time and search node is the worst among the three algorithms, but it can find the optimal path; The search time and search nodes of A* -Chebyshev distance algorithm are medium among the three algorithms, and the optimal path can be found, but the search nodes are significantly more than that of A* -Manhattan distance algorithm. Although the three algorithms have their own advantages, they are still unable to optimize node elimination and search efficiency well, so it is necessary to further optimize the A* algorithm.

3. Algorithm improvement

3.1 Improved A* algorithm

In order to further improve the search efficiency and the number of search nodes, the heuristic function is weighted. The weighting function is, and the expression of the evaluation function is [20]:

$$f(n) = g(n) + h(n) * w(n) \tag{7}$$

In formula (7), $w(n)$ represents the weight corresponding to the n node. In the process of search, when $h(n)$ is less than the actual cost, the smaller $h(n)$ is, the more nodes A* searches, the slower the search efficiency will be. When $h(n)$ is equal to the actual cost, the A* algorithm only looks for the optimal path and does not search for other nodes. When $h(n)$ is greater than the actual cost, the A* algorithm can not guarantee to find an optimal path by expanding the search node. The larger the weight $w(n)$, the A* algorithm will preferentially expand to the end point, and the search speed will be accelerated, but the local optimal path may appear. The smaller the weight $w(n)$ is, the A* algorithm preferentially searches the optimal path, which is bound to increase the search nodes and slow down the search speed. A good search algorithm should not only consider the planned path, but also consider the speed of the search. Based on this consideration, dynamic weighting method is set up to solve the above problems.

After studying the distance algorithm of A* algorithm, this paper adopts the fusion of A* -Manhattan distance algorithm and A*-Chebyshev distance algorithm, and realizes the complementary advantages of the algorithm and optimizes the node and search efficiency through dynamic weighting. The improved distance strategy is as follows:

$$d = \alpha d_M + \beta d_C \tag{8}$$

Where, d is the distance calculation method adopted in this paper, d_M is the Manhattan distance between the current node and the target point, d_C is the Chebyshev distance between the current node and the target point,α and βis the dynamic weight factor.

$$w(n) = (1 + d / S) \tag{9}$$

$$S = (x_g - x_c) * (y_g - y_c) \tag{10}$$

In formula (10), (x_g, y_g) is the horizontal and vertical coordinate of the target point, and (x_c, y_c) is the current horizontal and vertical coordinate of the robot's location.

In order to verify the effectiveness of the algorithm, the simulation of the improved A* algorithm is shown in Figure 4, and the parameter table is shown in Table 3.

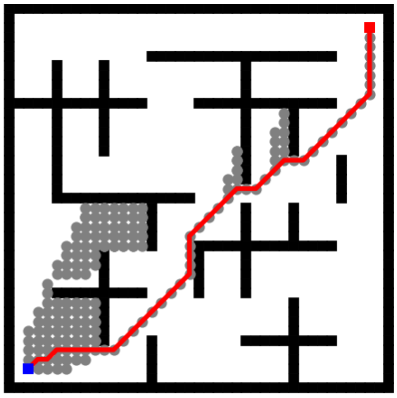


Figure 4: Improved A* simulation diagram

Table 3: Improved A* parameter table

Algorithm	Simulation time (s)	Number of traversal nodes	Planning distance
Improves the algorithm	0.769	163	57.355

Compared with Dijkstra's algorithm, the improved A* algorithm in this paper reduces the simulation time and the number of traversing nodes by 79.56% and 83.63% respectively. The A*-Hamilton distance cannot find the optimal path, so it is not considered. Compared with A*-Euclidean distance, the simulation time and the number of traversing nodes are reduced by 70.17% and 68.16% respectively. Compared with A*-Chebyshev distance, the simulation time and number of traversing nodes are reduced by 52.88% and 56.30%, respectively. The improved A* algorithm can find the optimal path, and the simulation time and the number of traversing nodes have obvious advantages compared with other algorithms.

3.2 Path optimization

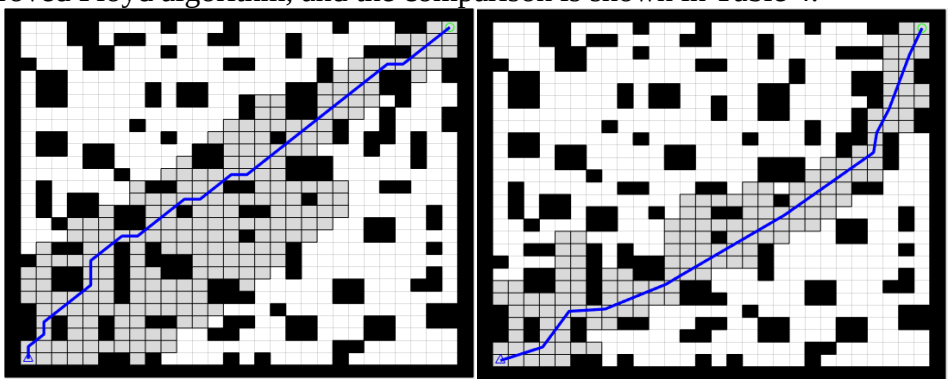
According to the motion of the robot in the real environment, the Angle path is not conducive to the movement of the robot, and it needs to be smoothed. Smoothing processing can find a path with the highest feasibility according to the actual path, and the smoothed path is more suitable for robot movement, and the robot can accurately control the turn and improve the controllability of the robot path [21]. This paper optimizes the global planning path based on the Floyd algorithm, the steps are as follows:

Step 1: Initialization, traverses all key nodes in A*. If the current node is collinear with multiple adjacent nodes, the previous node of the current node is a redundant node and node judgment is performed.

Step 2: After deleting redundant nodes, plan a path in advance and check whether the safe distance between the path and obstacles is greater than the threshold. If the distance is greater than the threshold, keep the planned path. If the value is smaller than the value, the planned path is close to the obstacle. Therefore, the planned path is discarded and the abandoned node is restored. The optimal smooth path is found through node path loop in turn.

Floyd algorithm optimization has the disadvantages of high time complexity in planning and is not suitable for calculating a large amount of data. In this paper, the node strategy is improved, and the key node of A* algorithm is preferentially selected as the traversal node of Floyd algorithm, which effectively guarantees the time complexity of Floyd algorithm. Considering that in an environment with a long planned path, the distance between two nodes is long, which is not

conducive to path planning, this paper uses the Floyd algorithm to dynamically increase the number of planned key nodes when the planned path is larger than the set value to prevent a large deviation between the planned path and the actual path due to a small number of key nodes. The improved Floyd algorithm is simulated to verify the feasibility of the algorithm. The improved Floyd algorithm was simulated to verify the feasibility of the algorithm, as shown in Figure 5. Figure 5-a shows the path planning after the improved A* algorithm, and Figure 5-b shows the path planning after the improved Floyd algorithm, and the comparison is shown in Table 4.



(a) (b)

Figure 5: Algorithm simulation diagram

Table 4: Algorithm comparison table

Algorithm	Number of turning points	Degree of inflection	Number of traversal nodes	Planning distance
Improve the A * algorithm	13	585.0	229	40.526
Improving the A * - Floyd algorithm	10	197.272	151	40.670

In this paper, the path smoothing of Floyd algorithm is improved on the basis of the improved A* algorithm. Simulation shows that the turn times of the improved smooth path are reduced by 23.07%, the turn Angle is reduced by 66.28% and the number of traversing nodes is reduced by 34.06%, all of which are greatly reduced. However, in the smoothing process, the smoothed path only increases by 0.35%.

4. Improve DWA algorithm

A* algorithm can only complete global path planning in A static environment. For a complex dynamic unknown environment, A* algorithm cannot properly plan the path, so it is necessary to adopt appropriate local path planning according to the specific environment [22]. The principle of DWA algorithm is to establish a kinematic model of linear velocity and angular velocity, fully consider the limitations and constraints of physical attributes such as velocity and acceleration in robot motion, and conduct multiple groups of samples in the velocity space to simulate a cluster of motion trajectories of multiple groups of velocity within a certain period of time, and then select the optimal trajectory and velocity through the evaluation function. Robot local path planning [23] is realized.

1) Robot kinematics model

The patrol robot adopts differential motion, which stipulates that the robot has only the linear velocity in the X direction and the rotational angular velocity in the Z axis. In a short period of time,

the robot movement can be regarded as a straight line movement. Assuming a short period of time, linear motion at a constant speed is established, and the kinematic model is expressed as [24]:

$$\begin{cases} x_{t+1} = x_t + v\Delta t cos\alpha_t \\ y_{t+1} = y_t + v\Delta t sin\alpha_t \\ \alpha_{t+1} = \alpha_t + w\Delta t \end{cases} \tag{11}$$

In formula 11, x_t, y_t, a_t and are the current position and direction Angle of the robot; x_{t+1}, y_{t+1}, a_{t+1} is the position and direction Angle of the robot at the next moment. v and w are the linear and angular velocity at time t. The formula shows that in a certain time window, the pose of the robot is determined by the current speed and heading Angle together, because different combinations of speed and heading Angle can produce different robot poses and form different motion trajectories. Since the speed and heading Angle are not constrained by any conditions, this combination can reach countless kinds, which is obviously not the goal of the design, and speed constraints are needed to optimize the robot movement.

2) Velocity sampling

Velocity sampling is to control the sampling interval within a certain range, so as to generate several limited combinations of velocity and acceleration, and establish the robot constraint conditions [25].

(1) Velocity and angular velocity constraints

According to the robot's own conditions and moving environment, the minimum, maximum speed and angular speed constraints are established. The expression is as follows:

$$\begin{cases} v_{min} \le v \le v_{max} \\ w_{min} \le w \le w_{max} \end{cases} \tag{12}$$

In formula 12, where, v_{min}, v_{max} is the minimum and maximum velocity, w_{min}, w_{max} is the minimum and maximum angular velocity.

(2) Acceleration and deceleration constraints

Due to the influence of motor performance, the maximum acceleration and deceleration speed of the robot is limited. The acceleration and deceleration constraint conditions of the robot are established, and the expression is as follows:

$$\begin{cases} v_c - a_{min}\Delta t \le v \le v_c + a_{max}\Delta t \\ w_c - a_{min}\Delta t \le w \le w_c + a_{max}\Delta t \end{cases} \tag{13}$$

In Formula 13, v_c, w_c, a_{min}, a_{max} respectively represent the current velocity, current angular velocity, maximum robot deceleration and maximum acceleration.

(3) Braking distance constraints

Based on the safety of robot movement, when the robot moves to obstacle 3, it is necessary to stop the robot movement within a certain distance and complete the brake to avoid collision between the robot and obstacle. The constraint condition of robot braking distance is established, and the expression is as follows:

$$v \le \sqrt{2dist(v,w) * a_{min}} \tag{14}$$

In Formula 14, $dist(v,w)$ is the minimum distance between the robot's trajectory and the obstacle when the velocity combination is (v,w).

3) Traditional evaluation function

Several feasible velocity combination trajectories are obtained by velocity sampling. This cluster of trajectories is scored by evaluation function, the optimal path is selected, and the evaluation function equation is established as:

$$G(v,w)=\sigma\begin{pmatrix}\alpha head(v,w)+\beta dis(v,w)\\+\gamma vel(v,w)\end{pmatrix} \tag{15}$$

In formula 15, σ is the smoothing function, α , β , γ is the weighting factor of each function. $head(v,w)$ is the heading Angle evaluation function, which is the Angle between the position of the robot and the target point under the current velocity combination. The smaller the Angle value, the higher the azimuth Angle evaluation function value, and the better the motion trajectory corresponding to the velocity combination. $dist(v,w)$ is the obstacle distance evaluation function, where $dist$ is the minimum distance between the trajectory curve corresponding to velocity combination (v,w) and the obstacle vehicle. The larger the value, the higher the value of the evaluation function, indicating that the motion trajectory corresponding to this velocity combination is better. $vel(v,w)$ is the velocity evaluation function, the closer the robot is to the target speed during normal driving, the better.

4) Improved evaluation function

Evaluation function is the decisive factor in robot path selection. The traditional DWA algorithm evaluates the path trajectory based on the local path through the evaluation function of heading Angle, obstacle distance and velocity, ignoring the guiding role of the global path to the robot. In addition, the traditional DWA algorithm only considers the distance between the robot and static obstacles. The dynamic influence of the dynamic obstacles on the robot in the location environment is not considered. Based on this, this paper introduces the dynamic obstacle distance evaluation function and the global path evaluation function to improve the robot's path selection. The improved evaluation function is as follows:

$$G(v,w)=\sigma\begin{pmatrix}\alpha head(v,w)+\beta dis(v,w)\\+\gamma vel(v,w)+\mu dis\_D(v,w)\\+\delta path(v,w)\end{pmatrix} \tag{16}$$

In formula 16, compared with formula 15, weight factors δ and μ are added, and global path evaluation function $path(v,w)$ is added. The closer the robot's motion trajectory is to the global path, the better the planned path will be. The dynamic obstacle distance evaluation function $dis\_D(v,w)$ enhances the robot's obstacle avoidance ability to unknown dynamic obstacles.

5) Normalization

In order to unify the ratio advantage of each evaluation function and make the evaluation function without comparability comparable [25], the evaluation function is normalized and the normalization equation is shown in formula 17:

$$\begin{cases} normal\_head(i) = \dfrac{head(i)}{\sum_{i=1}^{n} head(i)} \\ normal\_dis(i) = \dfrac{dis(i)}{\sum_{i=1}^{n} dis(i)} \\ normal\_vel(i) = \dfrac{vel(i)}{\sum_{i=1}^{n} vel(i)} \\ normal\_dis\_D(i) = \dfrac{dis\_D(i)}{\sum_{i=1}^{n} dis\_D(i)} \\ normal\_path(i) = \dfrac{path(i)}{\sum_{i=1}^{n} path(i)} \end{cases} \tag{17}$$

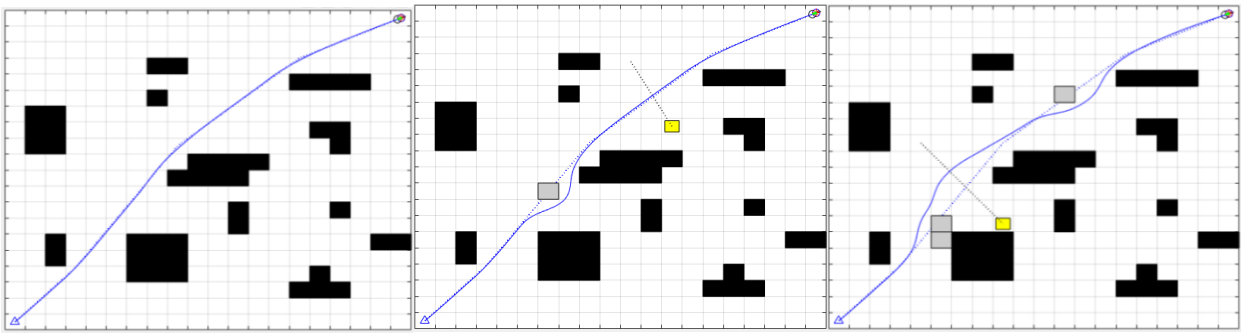
5. Algorithm fusion simulation and implementation

In this paper, the improved A* algorithm is used for global path planning and the improved DWA algorithm for local path planning. The improved A* algorithm can accelerate the search speed, improve the search efficiency, and accurately find the optimal path. However, as A global path planning, A* algorithm ignores the influence of dynamic environment on path planning, and cannot meet the requirements of local obstacle and dynamic obstacle path planning. DWA algorithm can satisfy the dynamic local path planning, but DWA algorithm is based on local planning method may lead to local optimization of the path instead of global optimization, thus limiting the navigation ability of the robot. Through the combination of global optimization and local real-time, the patrol robot can intelligently plan the path in the complex environment and respond to the change of the environment in time. In this paper, A path search algorithm combining improved A* algorithm and improved DWA algorithm is used to verify the feasibility and superiority of the algorithm through simulation experiments.

Simulation experiments include simple map simulation and complex map simulation. The simulation of no unknown obstacle, single unknown static obstacle and dynamic obstacle, and multiple static obstacle and dynamic obstacle are respectively simulated, as shown in Figure 6 and Figure 7. Where black is the static obstacle, white is the free area, gray is the unknown static obstacle, yellow is the dynamic obstacle, blue triangle is the starting point, blue circle is the target point, and blue line is the planned path.

1) Simple map simulation

Figure 6-a shows the simulation of no unknown obstacles by the fusion algorithm, Figure 6-b show the simulation of a single unknown static obstacle and dynamic obstacle by the fusion algorithm, and Figure 6-c show the simulation of multiple static and dynamic obstacles by the fusion algorithm.

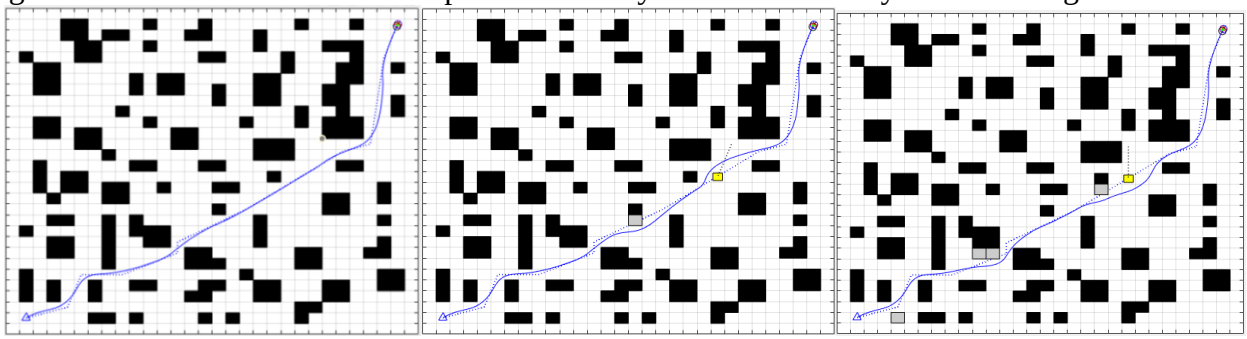


(a) (b) (c)

Figure 6: Simple map simulation

2) Complex map simulation

Figure 7-a is the simulation of no unknown obstacles by the fusion algorithm, Figure 7-b is the simulation of a single unknown static obstacle and dynamic obstacle by the fusion algorithm, and Figure 7-c is the simulation of multiple static and dynamic obstacles by the fusion algorithm.



(a) (b) (c)

Figure 7: Complex map simulation

Through the fusion algorithm simulation of different environments, the feasibility of the algorithm is verified. The robot can normally avoid static obstacles and dynamic obstacles, and path planning can realize path planning, which meets the continuous control requirements of the robot movement, meets the precise control requirements of the robot, and meets the requirements of the expected movement.

6. Conclusion

In this paper, A multi-algorithm fusion path planning algorithm for patrol robots is proposed, which optimizes the A* algorithm by searching the neighborhood and heuristic function. Under the condition of ensuring the optimal global path, the number of ubiquitous nodes and simulation time are reduced to varying degrees, avoiding the expansion of redundant nodes and improving the search efficiency of the algorithm. The improved A* algorithm still has node redundancy, excessive path transition and other phenomena. Floyd algorithm is used to introduce improved A* key nodes to optimize the improved A* algorithm again, eliminate redundant nodes, smooth the global path, and dynamically increase the number of key nodes for long-distance key nodes to effectively prevent path deviation. In view of the shortcomings of the improved A* algorithm in dynamic obstacle planning, the improved DWA algorithm is integrated to achieve local path planning, and the integrated path planning algorithm has local dynamic and unknown environment obstacle avoidance. Finally, the design algorithm is tested in real environment through the patrol robot platform. The experiment shows that the fusion algorithm proposed in this paper has the ability of

global path planning and local path planning, and verifies the feasibility of the fusion algorithm.

Acknowledgements

Research project "Research on Industrial Robot Fault Diagnosis Technology Based on Industrial Internet", project No. 2023AH052918, supported by "Research Center for New Energy Vehicle Motor Drive Technology of Anhui Institute of Information Engineering's School level Science and Technology Innovation Platform", "Research Team for Automotive Electronic Control System of Anhui Institute of Information Engineering's School level Scientific Research Team.

References

[1] Dudeja C, Kumar P. An improved weighted sum-fuzzy Dijkstra's algorithm for shortest path problem (iWSFDA) [J]. Soft Computing, 2022, 26(7): 647-676.
[2] Tang G, Tang C, Claramunt C, et al. Geometric A-star algorithm: An improved A-star algorithm for AGV path planning in a port environment[J]. IEEEAccess, 2021, 9:59196-59210.
*[3] Zheng Chuanchuan, Ke Fuyang, Tang Qinqin. Research on Autonomous Navigation Simulation by Integrating Improved D * and Gmapping Algorithms [J]. Computer Simulation, 2023, 40(10): 452-457+518.*
*[4] Lin Yifan, Chen Yanjie, He Bingwei, et al. Motion planning method for mobile robots without collision detection RRT *[J]. Journal of Instrumentation, 2020, 41(10): 257-267.*
[5] Lei Wu, Xiaodong Huang, Junguo Cui, et al. Modified adaptive ant colony optimization algorithm and its application for solving path planning of mobile robot[J]. Expert Systems with Applications, 2022, 215(1): 119410.
[6] Huang Rongjie, Wang Yagang. Smooth Path Planning for Robots Based on Viewable and Improved Genetic Algorithm [J]. Control Engineering, 2024, 31(04): 678-686.
[7] Chen Qiulian, Zheng Yijun, Jiang Huanyu, et al. Dynamic Path Planning Based on Neural Network Improved Particle Swarm Optimization [J]. Journal of Huazhong University of Science and Technology (Natural Science Edition), 2021, 49(02): 51-55.
[8] Xu Wan, Cheng Zhao, Zhu Li, et al. A Local Path Planning Algorithm Based on Improved Artificial Potential Field Method [J]. Electronic Measurement Technology, 2022, 45(19): 83-88.
[9] Haixu Yang; Xiaoming Xu; Jichao Hong. Automatic Parking Path Planning of Tracked Vehicle Based on Improved A and DWA Algorithms [J]. IEEE, 2022, 9(1): 283-292.*
*[10] Guo Zhijun, Yin Yakun, Li Yixuan, et al. Path planning for mobile robots by integrating improved A * and TEB algorithms [J]. Journal of Henan University of Science and Technology (Natural Science Edition), 2023, 44(04): 57-65+7.*
[11] TANG Zhuozhen, MA Hongzhong. An Overview of Path Planning Algorithms[J]. Iop Conference Series: Earth and Environmental Science, 2021, 804(2):022024.
*[12] Hu Zheng, Xu Bin. Dynamic Path Planning Integrating A * Algorithm and Artificial Potential Field Method [J]. Combination Machine Tool and Automation Processing Technology, 2023(07): 46-49+56.*
[13] Yin X, Cai P, Zhao K, Zhang Y, et al. Dynamic path planning of AGV based on kinematical constraint A algorithm and following DWA fusion algorithms[J]. Sensors, 2023, 23(8):4102.*
[14] Li Sanping, Yuan Longqiang, Wu Liguo, et al. Path planning for mobile robots based on improved fusion ant colony algorithm [J]. Mechanical design, 2023, 40(10): 76-84.
[15] Xiang D, Lin H, O Y J, et al. Combined improved A and greedy algorithm for path planning of multi-objective mobile robot[J]. Scientific Reports, 2022, 12(1):13273.*
[16] Chao Liu, Lei Wu, Guangxin Li, et al. Improved multi-search strategy A algorithm to solve three-dimensional pipe routing design[J]. Expert Systems with Applications, 2024, 240: 122313.*
[17] Li C, Huang X, Ding J. Global path planning based on a bidirectional alternating search A algorithm for mobile robots[J]. Computers & Industrial Engineering, 2022, 168:108123.*
[18] Jiang H, Sun Y. Research on Global Path Planning of Electric Disinfection Vehicle Based on Improved A Algorithm [J]. Energy Reports, 2021, 7:1270-1279.*
[19] Chao Liu, Lei Wu, Wensheng Xiao. An improved heuristic mechanism ant colony optimization algorithm for solving path planning [J]. Knowledge-based systems, 2023, 271:110540.
*[20] Yang Fangqing, Liu Jicheng. Path Planning for Mobile Robots by Integrating Improved A * Algorithm and Dynamic Window Method [J]. Industrial Control Computer, 2021, 34(5):106-108, 112.*
[21] Bian Yongming, Ji Pengcheng, Zhou Yihe, et a. Obstacle avoidance path planning for mobile robots based on improved DWA [J]. Chinese Journal of Engineering Machinery, 2021, 19(1):44-49.

*[22] Yang Guihua, Wei Jiale. Logistics robot path planning based on improved A * and DWA algorithms [J]. Science, Technology and Engineering, 2022, 22(34):15213-15220.*
*[23] Wang Hongbin, Yin Pengheng, Zheng Wei, et al. Path planning for mobile robots based on improved A * algorithm and dynamic window method [J]. Robot, 2020, 42(3): 346-353.*
[24] Te Wang, Aijuan Li, Dongjin Guo, et al. Global Dynamic Path Planning of AGV Based on Fusion of Improved A Algorithm and Dynamic Window Method[J]. Sensors, 2024, 24(6): 2011.*
*[25] Liu Zhouhao, Wan Chaoyi, Yin Mingfeng, et al. Mobile robot path planning using improved A * algorithm and improved DWA algorithm [J]. Manufacturing automation, 2023, 45(12): 55-60.*