International Journal of Control, Automation, and Systems 22(4) (2024) 1385-1399
http://dx.doi.org/10.1007/s12555-022-1016-5
ISSN:1598-6446 eISSN:2005-4092
http://www.springer.com/12555

A Real-time Path Planning Algorithm for Mobile Robots Based on Safety Distance Matrix and Adaptive Weight Adjustment Strategy

Xinpeng Zhai, Jianyan Tian*, and Jifu Li

Abstract: The fusion of the A* and the dynamic windowing algorithm is commonly used for the path planning of mobile robots in dynamic environments. However, the planned path has the problems of redundancy and low security. This paper proposes a path planning algorithm based on the safety distance matrix and adaptive weight adjustment strategy to address the above problems. Firstly, the safety distance matrix and new heuristic function are added to the traditional A* algorithm to improve the safety of global path. Secondly, the weight of the evaluation sub-function in the dynamic window algorithm is adjusted through an adaptive weight adjustment strategy to solve the problem of path redundancy. Then, the above two improved algorithms are fused to make the mobile robot have dynamic obstacle avoidance capability by constructing a new global path evaluation function. Finally, simulations are performed on grid maps, and the fusion algorithm is applied to the actual mobile robot path planning based on the ROS. Simulation and experimental results show that the fusion algorithm achieves optimization of path safety and length, enabling the robot to reach the end point safely with real-time dynamic obstacle avoidance capability.

Keywords: Adaptive weight, A* algorithm, dynamic window algorithm, path planning, safety distance matrix.

1. INTRODUCTION

Path planning is an important part of autonomous navigation technology for mobile robots, and the purpose is to plan a collision-free optimal or suboptimal path from the starting point to the end point [1-3]. According to the completely known, partially known, or completely unknown information about the environment where the mobile robot is located, path planning can be divided into global path planning and local path planning [4,5].

Global path planning algorithms include the Dijkstra algorithm [6], ant colony algorithm [7], fast random search tree algorithm (RRT) [8], A* algorithm [9], etc. The Dijkstra algorithm adopts the breadth-first search method to find the shortest path from the starting point to the endpoint by calculating the distance from the starting point to the other nodes. However, when the environment is more complex, the number of computing nodes increases, and the computing efficiency decreases [10]. The ant colony algorithm uses the mathematical model of ants cooperatively searching for food to obtain the optimal path, which has the advantages of being robust and easy to combine with other algorithms because of its distributed nature. However, it also has the disadvantages of slow convergence speed and easy to fall into local optimum [11]. The RRT is a sampling algorithm, which has fast search speed and strong searchability. However, the search accuracy is low and the path smoothness is poor [12]. The A* algorithm calculates the cost value of nodes by heuristic function to get the optimal extended node and finally generates the optimal path. However, the planned path has problems such as unsmooth, node redundancy, and low path security [13,14]. Compared with other global path planning algorithms, the A* algorithm is widely used in the autonomous navigation of mobile robots due to its high computing efficiency. In the autonomous navigation of mobile robots, the safety of the search path is the most important prerequisite for the implementation of all functions [15]. To improve the security of the search path planned by the A* algorithm, many scholars have improved the A* algorithm from different aspects. Zhang *et al.* [16] introduced the distance cost between nodes and obstacles to improve the safety of path planning in the A* algorithm. When there are many obstacles, it needs to calculate the distance between each obstacle and the current node, which reduces the search efficiency. Chen *et al.* [17] made the path planned by the A* algorithm away from obstacles through the shifting process, but the fixed shifting distance is not suitable for complex environments. Duan *et al.* [18] introduced a matrix in the A* algorithm to ensure safe dis-

Manuscript received October 27, 2022; revised March 16, 2023; accepted May 26, 2023. Recommended by Associate Editor Tae-Koo Kang under the direction of Senior Editor Euntai Kim. This work was supported by the Nature Scinece Foundation of Shanxi Province of China under Grant (201901D111092) and Key Research and Development (R&D) Projects (Intelligent Field) of Shanxi Province of China (2021_6)

Xinpeng Zhai, Jianyan Tian, and Jifu Li are with the College of Electrical and Power Engineering, Taiyuan University of Technology, Taiyuan 030024, China (e-mails: {zxp_ql, tut_tianjy}@163.com, lijifu@tyut.edu.cn).
* Corresponding author.

tances under different environments. However, it specifies that the mobile robot can only move in four directions, which does not meet the actual robot motion requirements. Besides, the expansion of obstacles can ensure the safety of path planning, but the expanded map will enlarge the inaccuracy when the created map is not accurate. Moreover, when the obstacle spacing is small, the expansion process will most likely cause the robot to be unable to follow the normal path, which may increase the length of the path and the search time.

In addition, although the above-improved method can ensure the safety of the path to some extent, it does not consider unknown obstacles and does not have dynamic obstacle avoidance capability. The combination of the global path algorithm and the local path algorithm enables the robot to dynamically avoid obstacles in real time based on the globally optimal path [19]. The local path planning methods include the artificial potential field method [20], neural network method [21], dynamic window method (DWA) [22], etc. The artificial potential field method constructs gravitational and repulsive fields around the target point and obstacles to plan the optimal path without collision according to the descent direction of the search potential function. When there are more obstacles in the environment, it will be multiple zero-potential points, and easy to fall into local optimum [23,24]. The neural network algorithm can quickly find the optimal solution under the training of a large number of sample data and is often used in local obstacle avoidance operations of mobile robots. However, it requires a large number of training samples and has high requirements on the neural network model [25]. The DWA algorithm combines sensing information to predict its trajectory at the next time interval from the linear and angular velocities collected in velocity space to achieve real-time obstacle avoidance. The DWA algorithm is widely used in mobile robot obstacle avoidance processing due to its low computational complexity, but it has the problem of redundant paths [26,27].

How to quickly and accurately plan a safe and effective path of travel in a complex and dynamic environment is of great research significance. To solve the problem of low path security of the A* algorithm and path redundancy of the DWA algorithm, a real-time path planning algorithm for mobile robots based on safety distance matrix and adaptive weight adjustment strategy is proposed. The safety distance matrix is added to the traditional A* algorithm to ensure the safety of its search. The heuristic function of the traditional A* algorithm is also improved to retain search efficiency. The weight of the evaluation sub-function in the dynamic window algorithm is adjusted through an adaptive weight adjustment strategy to solve the problem of redundant paths. Finally, the two improved algorithms are fused by constructing a global evaluation function to enable the robot to achieve dynamic real-time obstacle avoidance based on the global optimum. Simulations and experiments verify the feasibility of the proposed fusion algorithm in this paper.

The main contributions can be summarized: 1) We add the safety distance matrix to the A* algorithm and construct a new heuristic function to guarantee the safety of the path. It improves the search method from the algorithm itself and reduces the requirement of environmental map accuracy during the search. 2) We proposed an adaptive weight adjustment strategy to adjust the weights of the evaluation sub-functions in the DWA algorithm in real-time to solve the path redundancy problem. 3) We constructed global evaluation functions to fuse the two improved algorithms, which enables the mobile robot to avoid obstacles dynamically in real-time based on the globally optimal path.

The other parts of this paper are organized as follows: Section 2 proposed an improved A* algorithm with the safety distance matrix. Section 3 described the adaptive weight adjustment strategy. Section 4 introduced the fusion of two improved algorithms. Section 5 is the simulation and experimental. Section 6 is the conclusions.

2. IMPROVING THE A* ALGORITHM

In this section, we first describe the basic principles of the traditional A* algorithm and analyze the problems of the traditional A* algorithm. Then, we introduce the key turning node extraction strategy and the safety distance matrix in detail.

2.1. The traditional A* algorithm

A* algorithm is a path planning algorithm combining a heuristic search algorithm and a breadth-first algorithm, which is the most efficient direct search algorithm for solving optimal paths in static environments. The A* algorithm selects the search direction by a cost function. It starts from the starting point and expands to the surrounding node. The generation value of each surrounding node is calculated by the heuristic function. The node with the smallest generation value is selected as the next expansion node. This process is repeated until the endpoint is reached. In the search process, since each node on the path is the minimum cost node, the obtained path cost is also the minimum. The cost function of the A* algorithm can be calculated by

$$F(n) = G(n) + H(n), \tag{1}$$

where $F(n)$ is the cost function, $G(n)$ is the actual path cost from the start point to the current node, $H(n)$ is the estimated cost from the current node to the endpoint, and is also called the heuristic function. The heuristic function can control the speed and accuracy of the A* algorithm.

When calculating $G(n)$ and $H(n)$, to make it closer to the real distance, the eight-direction search and Euclidean

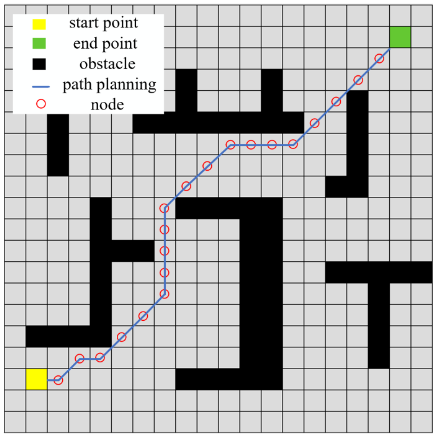


Fig. 1. The simulation result of traditional A* algorithm.

distance are used as the criteria for judging the cost. It can be calculated as

$$\begin{cases} G(n) = sqrt[(x_{curr} - x_{start})^2 + (y_{curr} - y_{start})^2], \\ H(n) = sqrt[(x_{curr} - x_{goal})^2 + (y_{curr} - y_{goal})^2], \end{cases} \quad (2)$$

where (x_{curr}, y_{curr}) is the coordinates of the current node, (x_{start}, y_{start}) is the coordinates of the start point, (x_{goal}, y_{goal}) is the coordinates of the endpoint.

The A* algorithm uses openlist and closelist to record node expansion and optimal node during the search process. When a node is searched, its surrounding nodes will be saved in the openlist. The cost value of these nodes is calculated according to formulas (1) and (2). The node with the smallest is selected as the next expansion node and added to the closelist at the same time. This process is repeated until the target point is added to the closelist and the A* algorithm is successful. The path planning result of the traditional A* algorithm is shown in Fig. 1.

As shown in Fig. 1, the path planned by the traditional A* algorithm contains many redundant nodes and unnecessary turning nodes. If there are too many turning nodes, the mobile robot can only move a short distance at a time, which will cause the robot to be stuck [28]. Besides, the robot is regarded as a point, and the safety distance between the robot and the obstacle is not considered. The path safety is low. It requires accurate environment map construction and high robot motion control accuracy. Based on the A* algorithm, the key fold extraction strategy and the security distance matrix are added to solve the above problems.

2.2. Key turning node extraction strategy

Aiming at the problem of many redundant nodes and unnecessary turning nodes in the traditional A* algorithm, this paper proposes a key turning nodes extraction strategy.The key turning node extraction strategy is as follows:

1) Get all node $U = \{P_i, 1 \leq i \leq n\}$ of the path planned by the A* algorithm, where P_1 is the starting point of the planned path, P_n is the end point. Create a turning node set V to store the extracted key turning node. The initial value of set V is $\{P_1, P_n\}$.
2) Use the area method to extract the turning node: Calculate the area enclosed by three adjacent nodes (P_{i-1}, P_i, P_{i+1}) in the path in turn. The area calculation formula can be expressed as

$$S = \sqrt{L(L - |P_{i-1}P_{i+1}|)(L - |P_{i-1}P_i|)(L - |P_iP_{i+1}|)}, \quad (3)$$

where S is the area of the triangle, L is the half-perimeter of a triangle, $|P_{i-1}P_{i+1}|$, $|P_{i-1}P_i|$, $|P_iP_{i+1}|$ is the side lengths of triangles. It can be calculated using (2). If $S = 0$, P_i is the redundant node. Otherwise, P_i is the turning node. Store the extracted turning nodes into the set V.
3) In the adjacent turning nodes (P_{k-1}, P_k, P_{k+1}) of the set V, connect two nodes P_{k-1} and P_{k+1} that are not adjacent. If the straight line of these two nodes passes through the obstacle area, the turning node P_k is a key turning node, otherwise it is a redundant turning node.

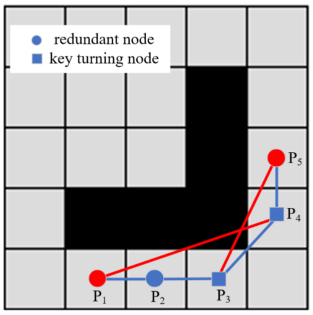


Fig. 2. Schematic diagram of key turning node extraction strategy.

The key turning node extraction strategy is described in detail in Fig. 2.

As shown in Fig. 2, $P_1 - P_5$ are the five nodes in the path. Because the nodes P_1, P_2, and P_3 are on a straight line, the area enclosed is 0, so P_2 is a redundant node. Connect node P_1 and node P_4, because the area of triangle $P_1P_3P_4$ is not 0, so P_3 is the turning node. The line segment P_2P_4 passes through the obstacle, so P_3 is the key turning node. Similarly, it can be judged that P_4 is also a key turning node.

The above method is used to remove redundant nodes and extract key turning node in Fig. 1. The result is shown in Fig. 3.

As shown in Fig. 3, after adding the key turning node extraction strategy, redundant nodes can be removed and key turning nodes can be extracted. During the actual navigation, the robot can move longer distances with a single command. It satisfies the non-holonomic constraints of mobile robots.

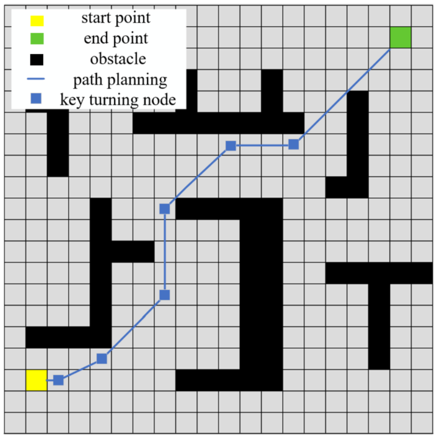


Fig. 3. The result of key turning node extraction.

2.3. The safe distance matrix

There is an appropriate safety distance between the robot and the obstacle, which is good protection for the robot and surrounding environment. The expansion of obstacles can improve path security, but it may increase the inaccuracy of maps [29,30]. Moreover, when the obstacle spacing is small, the expansion process will cause the robot unable to follow the normal path, which may increase the path length and search time [31].

In this paper, the safety distance matrix and matching heuristic function are proposed to improve the safety of the path. This method does not require an accurate environment map or reconstructed map. It is simple and easy to operate, which can better solve the problem of low path security.

The safety distance matrix proposed is different from the distance cost between nodes and obstacles, which refers to the distance cost of all obstacles to the node within a certain range [32]. The safety distance matrix can be regarded as a rectangular range with the current node at the center. It only needs to consider whether there are obstacles in the four corners of the rectangle. If there is an obstacle, the cost of the current point is increased, making the path away from the obstacle. The safety distance matrix can be expressed as

$$K_n = \begin{bmatrix} 2 & 0 & \cdots & \cdots & \cdots & 0 & 2 \\ 0 & 0 & \cdots & \cdots & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \iddots & \vdots & \vdots \\ \vdots & \vdots & \cdots & 1 & \cdots & \vdots & \vdots \\ \vdots & \vdots & \iddots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & \cdots & \cdots & 0 & 0 \\ 2 & 0 & \cdots & \cdots & \cdots & 0 & 2 \end{bmatrix}, \tag{4}$$

where $K_n \in R^{d\times d}$, the number 0 has no real meaning, the number 1 represents the next node the robot may reach, the number 2 indicates the four positions to be detected.

During the path search, in addition to calculating the cost at "1" in the safety distance matrix, it is also necessary to detect whether there is an obstacle at the position of "2". The cost function after adding the safety distance matrix can be calculated by

$$F'(n) = G(n) + H(n) + aS(n), \tag{5}$$

where $S(n)$ is the cost value of the safety distance matrix, a is the weight coefficient.

When an obstacle is detected at the position of "2" in K_n, the value of S_n is set to 1. When an obstacle is not detected at the position of "2" in K_n, the value of S_n is set to 0. If the value of a is much smaller than $G(n)$ and $H(n)$, the safety distance matrix may fail. If the value of a is much larger than $G(n)$ and $H(n)$, the planned path will be far away from the obstacles. The number of path search nodes will increase and reduce the efficiency of the search. It can be expressed as

$$a = \left[\sqrt{(M/L)^2 + (N/L)^2}\right], \tag{6}$$

where M, N is working range of the robot, L is the safety distance, [] is the rounding operation. When the robot is circular, the radius can be chosen as the safety distance. When the robot is rectangular, half of the longest side can be chosen as the safety distance.

The appropriate dimension of the safety distance matrix should be selected according to the size of the robot and the map resolution. It can be expressed as

$$d = [2 \times (L \times b) + 1], \tag{7}$$

where d is the dimension of the safety distance matrix, b is the map resolution.

To illustrate the safety distance matrix K_n more clearly, we calculated the cost values of the node involved in the path search. The path search cost is shown in Fig. 4.

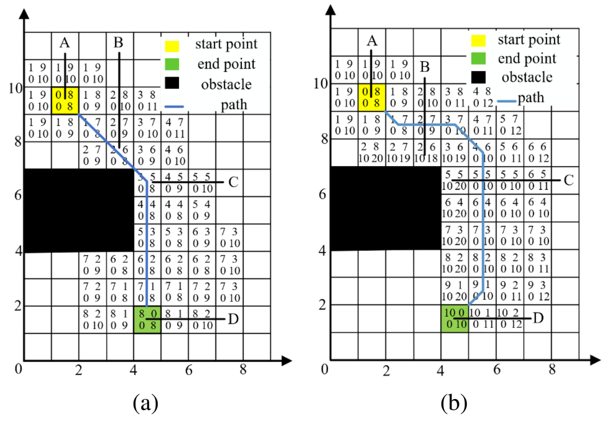


Fig. 4. The search cost value of path planning. (a) The search cost of traditional A*. (b) The search cost of adding the safety distance matrix.

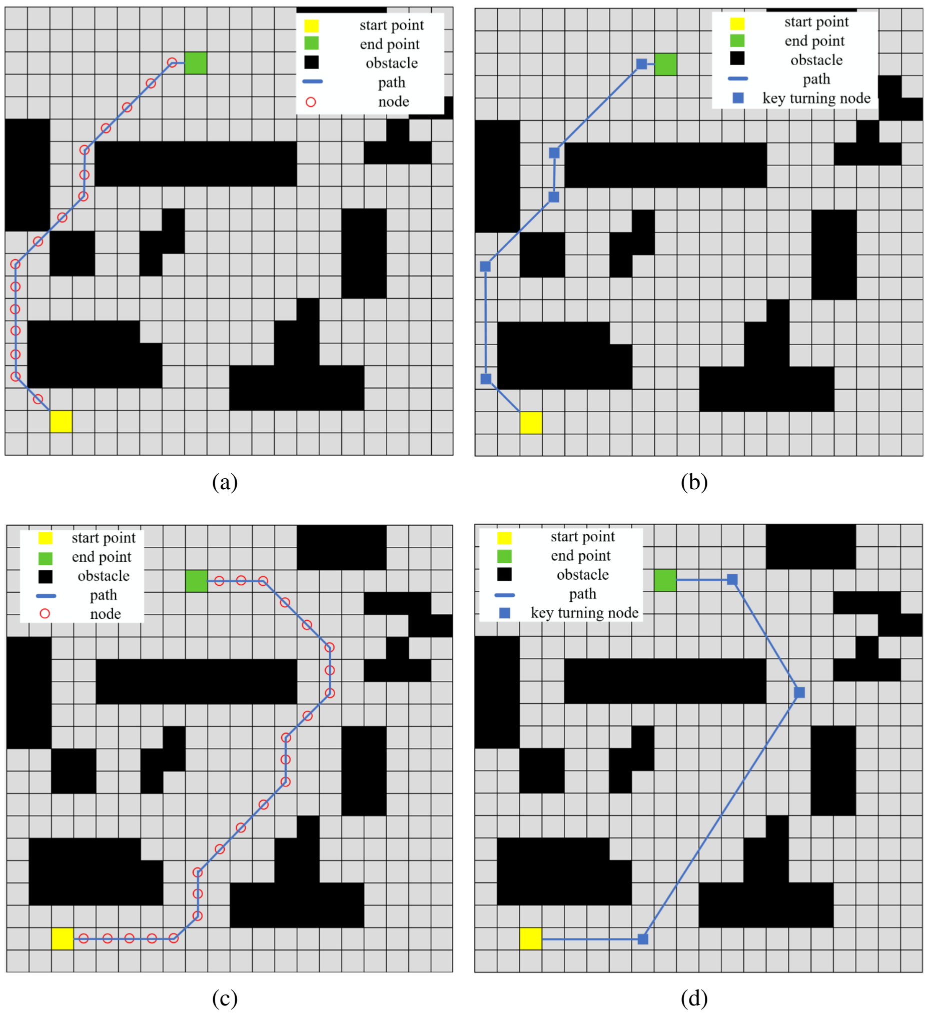


Fig. 5. The results of ablation experiments. (a) The path planning of traditional A*. (b) The result of adding key turning node extraction strategy. (c) The result of adding safety distance matrix. (d) The result of improved A*.

As shown in Fig. 4, the yellow grid represents the starting point, the green grid represents the end point, the black grid represents the obstacle. The grid without numbers indicates the nodes that are not related to the search process, and the grid with numbers is the node related to the search process. The four numbers in the grid with numbers represent $G(n)$, $H(n)$, $aS(n)$, and $F'(n)$ in formula (5), respectively. The value of a is 10, and the dimension of the safety distance matrix is 3. As can be seen from Fig. 4(a), the path (AC and CD) planned by the traditional A* algorithm is close to the obstacle, and the safety of the path is poor. When the environment map is inaccurate or the robot control has errors, the robot will be easily connected to the obstacle. It can be seen that after adding the safety distance matrix, the cost value $F'(n)$ of point B in Fig. 4(b) is much larger than the point B in Fig. 4(a). This is because the point below point B is at the "1" position in formula (4), and there is an obstacle at the "2" position, so the value of S_n is 1. According to formula (5), the cost value $F'(n)$ of point B in Fig. 4(b) is bigger than point B in Fig. 4(a). The algorithm will select the point with a smaller value of $F'(n)$ as the next position. After adding the safety distance matrix, the path planned by the algorithm will not pass through point B. Finally, it gets the path as shown in Fig. 4(b). The path is far away from obstacles, with good safety, and can effectively avoid the collision.

To verify the effectiveness of the safety distance matrix and key turning extraction strategy on the A* algorithm, we set up ablation experiments. The result is shown in Fig. 5. The grid map size is 20×20, and the length of the unit grid is 0.20 m.

As shown in Fig. 5(a), the path planned by the traditional A* algorithm is close to the obstacles, and the safety of the path is poor. If the environment map is not accurate or the robot control has errors, it is easy to cause the robot and the obstacles to collide. In addition, there are many redundant nodes in the path, and the motor needs to be started and stopped many times, which will cause damage

Table 1. Comparison result of traditional A* and improved A*.

	The path length (m)	DNO (m)	Search time (s)
Traditional A*	3.91	0	3.63
Improved A*	5.18	0.20	2.69

to the motor. As can be seen from Fig. 5(b), after adopting the key turning point extraction strategy, redundant nodes can be removed and key turning nodes can be extracted. It reduces the number of motor starts and stops, and reduces the energy consumption of the robot. As can be seen from Fig. 5(c), compared with the traditional A* algorithm, the path planned adding the safety distance matrix is far away from the obstacles, and the safety of the path is better. The requirements for the accuracy of the environment map or the robot control accuracy are not high, which can effectively avoid the collision between the robot and the obstacles. As shown in Fig. 5(d), the improved A* algorithm optimizes the path in terms of the number of nodes and path security. During the actual navigation, the robot can move longer distances along the safe path with a single command.

The path length, the distance to the nearest obstacle (DNO), and the path search time are used as performance indicators to quantitatively analyze the paths of Figs. 5(a) and 5(d). The results are shown in Table 1.

It can be seen from Table 1 that the improved A* algorithm increases the total path length by 1.27 m compared with the traditional A* algorithm, but increases the distance from the nearest obstacle by 0.20 m, which improves the security of the path. The search time of the proposed A* algorithm includes the path search time and other processes (removing redundant nodes and checking key turning nodes).The search time of the traditional A* algorithm is only the path search time. The improved A* algorithm

Table 2. Comparison result of RRT, Ant colony, and Improved A*.

Scenarios	Algorithm	The path length (m)	DNO (m)	Search time (s)
Scenario 1	RRT	11.48	0.1	4.26
	Ant colony	9.59	0	5.34
	Improved A*	9.37	0.14	5.09
Scenario 2	RRT	10.64	0.1	2.52
	Ant colony	13.58	0	3.56
	Improved A*	9.75	0.13	3.44

reduces the path planning time by 0.94 s compared to the traditional A* algorithm. The reason is that after adding the safety distance matrix to the traditional A* algorithm, the algorithm can sense the obstacles around the current node in advance.

We also compared the proposed A* algorithm with the RRT, and ant colony in different scenarios. The grid map size is 30×30, and the length of the unit grid is 0.20 m. The results are shown in Figs. 6 and 7.

As shown in Figs. 6 and 7, the coverage and complexity of the obstacles are different, and the maps are representative. The above 3 path-planning algorithms can plan a path from the starting point to the endpoint, but the path length, the distance to the nearest obstacle, and the search time are different. The initial parameters of 3 path-planning algorithms are set consistently. The algorithms are run 10 times and the average value is taken. The results are shown in Table 2.

As shown in Table 2, the RRT algorithm has the shortest search time, but the path length is the longest. Besides, it generates too many turning nodes. The robot will consume more time in actual movement. The improved A* algorithm is better than the ant colony algorithm in all as-

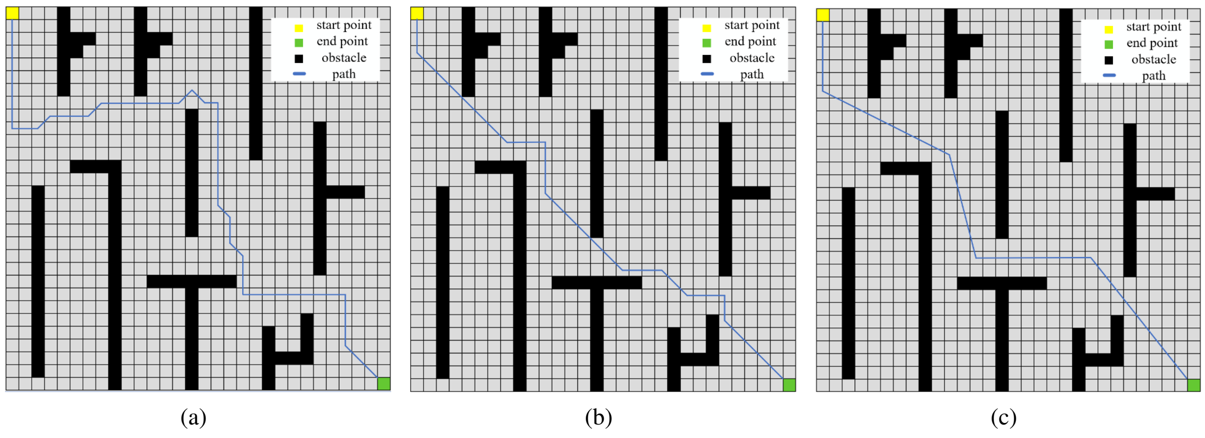


Fig. 6. The simulation results of scenario 1. (a) The path planning of RRT algorithm. (b) The path planning of ant colony algorithm. (c) The path planning of improved A* algorithm.

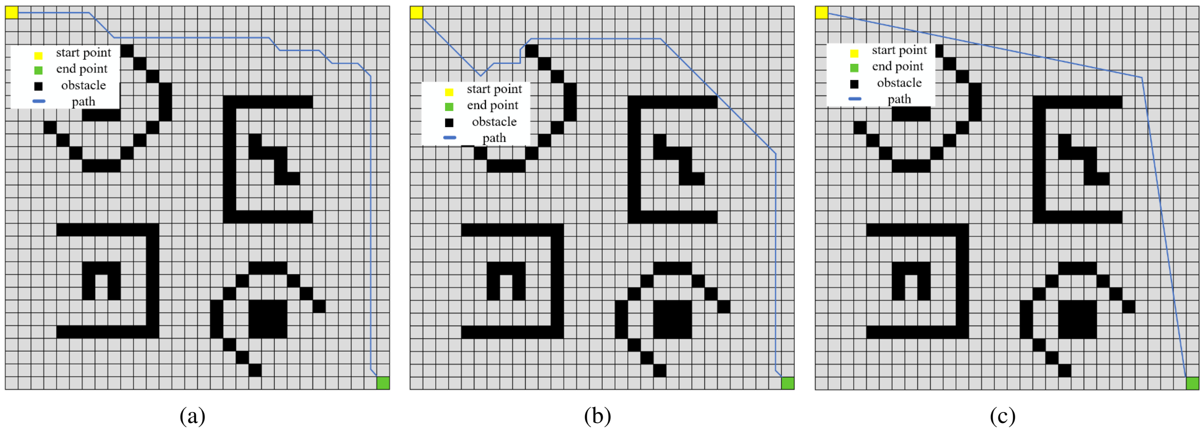


Fig. 7. The simulation results of scenario 2. (a) The path planning of RRT algorithm. (b) The path planning of ant colony algorithm. (c) The path planning of improved A* algorithm.

pects, and the planned path length is shorter and safer.

In this chapter, we introduce the construction of the safe distance matrix and how to choose the appropriate dimension. It should be noted that compared with map expansion method, the safety distance matrix proposed in this paper does not improve the safety of the path by changing the map, but by modifying the search methodology of the algorithm to improve safety of the path.

3. IMPROVED DWA ALGORITHM

Although the above improved A* algorithm can obtain the optimal safe path from the starting point to the ending point, it does not consider unknown obstacles. It cannot avoid dynamic obstacles. In this paper, an improved DWA algorithm is proposed to solve the above problem.

3.1. The traditional DWA algorithm

The traditional DWA algorithm samples the speed space $(v_t,\ \omega_t)$ of the mobile robot in the window area and then simulates the motion trajectory under this speed space. The changes of the linear velocity v_t and the angular velocity ω_t in the velocity space represent the motion state of the mobile robot. Among all feasible trajectories, the optimal trajectory is obtained through the evaluation function. Assuming that the mobile robot moves in a time interval Δt, the kinematic model of the mobile robot can be calculated by

$$\begin{cases} x_{t2} = x_{t1} + v_t \Delta t \cos(\Delta\theta), \\ y_{t2} = y_{t1} + v_t \Delta t \sin(\Delta\theta), \\ \theta_{t2} = \theta_{t1} + \omega_t \Delta t, \end{cases} \tag{8}$$

where $(x_{t2},\ y_{t2},\ \theta_{t2})$ is the pose of the robot in the world coordinates at time $t+\Delta t$, $(x_{t1},\ y_{t1},\ \theta_{t1})$ is the pose of the robot in the world coordinates at time t, $\Delta\theta = \omega_t \Delta t$.

There are infinitely many groups $(v_t,\ \omega_t)$ in the velocity space. In practice, it is necessary to constrain the sampling speed range according to the constraints of the mobile robot itself and the environment. The speed constraint of the mobile robot can be calculated by

$$V_m = \{(v,\omega) \mid v \in [v_{\min},\ v_{\max}],\ \omega \in [\omega_{\min},\ \omega_{\max}]\}. \tag{9}$$

The velocity constraint caused by the motor during the dynamic window movement interval of the mobile robot can be estimated by

$$V_d = \big\{(v,\omega)\ | v \in [v_c - \dot{v}_b \Delta t,\ v_c + \dot{v}_a \Delta t], \\ \omega \in [\omega_c - \dot{\omega}_b \Delta t,\ \omega_c + \dot{\omega}_a \Delta t]\big\}, \tag{10}$$

where $(v_c,\ \omega_c)$ is current speed of the mobile robot, $(\dot{v}_a,\ \dot{\omega}_a)$ is maximum acceleration of mobile robots, $(\dot{v}_b,\ \dot{\omega}_b)$ is minimum acceleration of mobile robots.

Mobile robot braking distance constraint: When avoiding obstacles in the local environment, the safety of the mobile robot needs to be ensured. Under the constraints of maximum deceleration, the speed can be reduced to 0 before a collision occurs. The braking constraint can be expressed as

$$V_b = \Big\{(v,\omega)\ | v \le (2d(v,\omega)\dot{v}_b)^{1/2}, \\ \omega \le (2d(v,\omega)\dot{\omega}_b)^{1/2}\Big\}, \tag{11}$$

where $d(v,\omega)$ is the closest distance between the track and the obstacle. This paper uses Euclidean distance to calculate it.

The sampling speed (v,ω) should satisfy the following formula

$$(v,\omega) \in \{V_m \cap V_d \cap V_b\}. \tag{12}$$

The predicted trajectory evaluation function of the DWA algorithm includes the heading angle evaluation sub-function, the speed evaluation sub-function, and the obstacle distance evaluation sub-function. It can be expressed as

$$Eval(v,\omega) = \alpha H(v,\omega) + \beta S(v,\omega) + \gamma D(v,\omega), \quad (13)$$

where $Eval(v,\omega)$ is the prediction trajectory evaluation sub-function, $H(v,\omega)$ is the heading angle evaluator sub-function, $S(v,\omega)$ is the speed evaluation sub-function, $D(v,\omega)$ is the obstacle distance evaluation sub-function, α, β, γ is the weight value corresponding to each evaluation sub-function.

The DWA algorithm gives priority to ensuring the security of the path, so it defines $\beta < \alpha < \gamma$.

Each sub-function needs to be normalized before calculating the predicted trajectory evaluation function. The normalization method is divided into I) and II) [33]. I) The current evaluation sub-function is divided by the sum of all evaluation sub-functions. II) 1 minus the current evaluator function divided by the sum of all evaluator functions. When the value of the evaluation sub-function is smaller and the score is higher, normalization should be performed using II). When the value of the evaluation sub-function is larger and the score is higher, normalization should be performed using I).

The prediction trajectory evaluation sub-function can be expressed as

$$H(v,\omega) = |\varphi - \theta|, \quad (14)$$

where φ is the angle between the line (connecting the current position of the robot and the target point) and the x-axis, θ is the heading angle. The value of $H(v,\omega)$ is larger, the score is higher. It should be normalized using I).

The heading angle evaluator sub-function can be expressed as

$$S(v,\omega) = |v_t|. \quad (15)$$

The value of $S(v,\omega)$ is larger, the score is higher. It should be normalized using I).

The obstacle distance evaluation sub-function can be expressed as

$$D(v,\omega) = d(P,O), \quad (16)$$

where P is the predicted position of the robot for time t, $O \in \{ob_1, ob_2, ob_3, \cdots, ob_i\}$ is the coordinates of the closest obstacle to the robot. The value of $SD(v,\omega)$ is smaller, the score is higher.It should be normalized using II).

To ensure the safety of the path, the traditional DWA algorithm defines the weight of the obstacle distance evaluation sub-function is always greater than the weight of other evaluation sub-functions. This weight rule causes the DWA algorithm to have a redundant route. This paper proposes a weight adaptive adjustment method to solve this problem.

3.2. DWA adaptive weight adjustment strategy

The weight adaptive adjustment strategy rules are as follows. when the environment of the mobile robot meets one or more of the following three conditions, the relationship between α and γ is set to $\alpha > \gamma$. The above three conditions are as follows:

1) When the distance between all obstacles (ob_i) and the current position (P) of the robot satisfies $Dist(P, ob_i) > \frac{3\sqrt{2}L}{2}$, we set $\alpha > \gamma$. That is, there are no obstacles in the area of radius $\frac{3\sqrt{2}L}{2}$ from the robot.
2) Assume that the current position of the robot is $P(x_p, y_p)$, the target point is $G(x_G, y_G)$, and the obstacle between the current position and the target point can be expressed as

$$ob_i(x_{ob_i}, y_{ob_i}) \in \{(x,y) \mid x_p \le x \le x_G,\ y_p \le y \le y_G\}. \quad (17)$$

When the line connecting the robot's current position and the target point does not pass through the obstacle area, we set $\alpha > \gamma$. It can be expressed by the following formula

$$L_{Dist}(ob_i, L_{P\to G}(ob_i)) > \frac{\sqrt{2}L}{2}, \quad (18)$$

where $L_{P\to G}$ is the connecting line segment between the robot's current position and the target point. It can be calculated by

$$\begin{cases} p_0 = (x,y);\ p_1 = (x_1,y_1);\ p_2 = (x_2,y_2); \\ L_{p_1\to p_2}(p_0) = (y_1-y_2)x + (x_2-x_1)y \\ \qquad\qquad + (y_2-y_1)x_1 + (x_1-x_2)y_1, \\ L_{Dist}(p_0, L_{p_1\to p_2}(p_0)) = \dfrac{|L_{p_1\to p_2}(p_0)|}{\sqrt{(x_2-x_1)^2+(y_1-y_2)^2}}. \end{cases} \quad (19)$$

3) When the robot is relatively close to the target point, we set $\alpha > \gamma$. It can be expressed by the following formula

$$Dist(P,G) \le \frac{\sqrt{2}L}{2}. \quad (20)$$

In order to verify that the DWA adaptive weight adjustment strategy can solve the problem of redundant routes. We perform simulation validation in a raster map with a grid size of 20×20 and a grid unit length of 0.2 m. The results are shown in Fig. 8.

As shown in Figs. 8(a) and 8(b), the starting point and end point are consistent and both algorithms can get the robot from the starting point to the ending point. The path length planned by the traditional DWA algorithm is 6.25

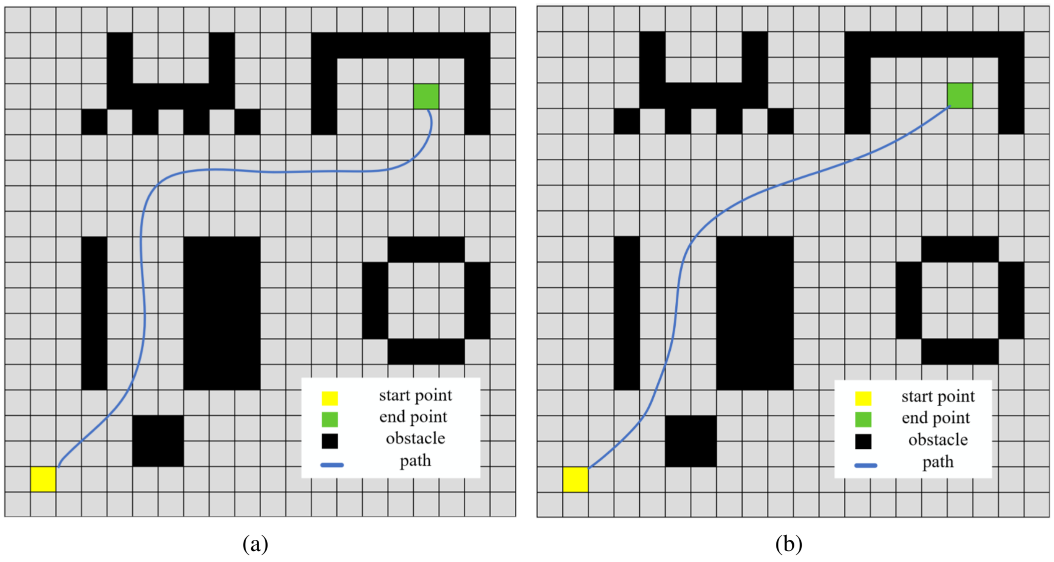


Fig. 8. The simulation results of traditional DWA and improved DWA. (a) The path planning of traditional DWA algorithm. (b) The path planning of improved DWA algorithm.

m. The path length planned by the improved DWA algorithm is 5.37 m. Compared with the traditional DWA algorithm, the path length planned by the improved DWA algorithm is shortened by 0.88 m and optimized by 14.1%. The reason for reducing path redundancy is that when there are no obstacles around the robot or when the robot is about to reach the end point, the adaptive weight adjustment strategy will automatically adjust the weights of the heading angle evaluation sub-function and the obstacle distance evaluation sub-function.

4. FUSION ALGORITHM

The traditional A* algorithm and DWA algorithm are improved respectively, but the path planned by improved A* algorithm does not have the ability of real-time dynamic obstacle avoidance. The path planned by the improved DWA algorithm is not the global optimal path. Therefore, this paper proposes a fusion algorithm combining the advantages of the two optimization algorithms. The flow chart of the algorithm is shown in Fig. 9.

As shown in Fig. 9, the static global path is first obtained by optimizing the A* algorithm, and the key turning points of the global path are extracted. Then, the key turning points other than the starting point are set as the local target points of the DWA algorithm in turn, and real-time local planning is performed until the target point is reached.

In order to make the local path planned by the DWA algorithm closer to the global path, the cost function of the A* algorithm is added to the predicted trajectory evaluation function as the global path evaluation sub-function.

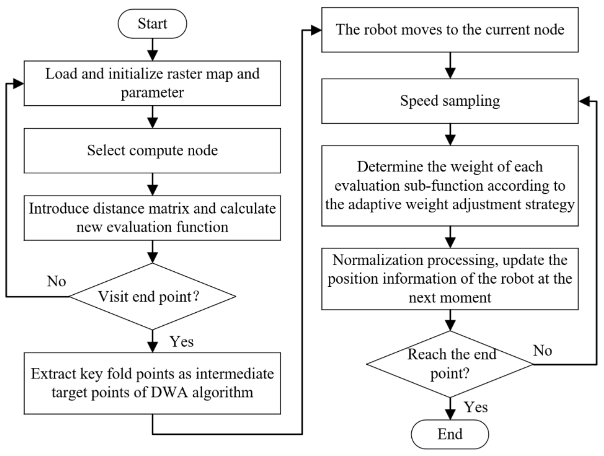

Start
Load and initialize raster map and parameter
Select compute node
Introduce distance matrix and calculate new evaluation function
Visit end point?
No
Yes
Extract key fold points as intermediate target points of DWA algorithm
The robot moves to the current node
Speed sampling
Determine the weight of each evaluation sub-function according to the adaptive weight adjustment strategy
Normalization processing, update the position information of the robot at the next moment
Reach the end point?
No
Yes
End

Fig. 9. The flow chart of the fusion algorithm.

The evaluation function of the fusion algorithm can be calculated by

$$\begin{cases} Eval'(v,\omega) = \alpha H'(v,\omega) + \beta S'(v,\omega) + \gamma D'(v,\omega) \\ \qquad + \eta G'(v,\omega), \\ G'(v,\omega) = F(n) = G(n) + H(n) + aS_n, \end{cases} \tag{21}$$

where η is the weight of global path evaluation, the value of η is less than α and γ, and greater than β. The value of $G(v,\omega)$ is smaller, the score is higher. It should be normalized using II). After normalizing each evaluation sub-function, it is brought into formula (20) to calculate a new evaluation sub-function.

5. SIMULATION AND EXPERIMENTAL ANALYSIS

5.1. The comparison experiment of different fusion algorithm

In order to verify the feasibility, applicability and effectiveness of the fusion algorithm proposed in this paper, simulation comparison experiments are conducted for the fused A* and DWA algorithms (A*-DWA) proposed in [15], the fused ant colony and artificial potential field algorithms (AC-APF) proposed in [20] and the fusion algorithm proposed in this paper, respectively. The experimental environment is a 64-bit Win 11 operating system running with 16GB of memory, the simulation platform is Python 3.7. The grid map size is 20×20, and the grid unit length is 0.2 m. A dynamic obstacle is randomly added in the simulation environment to verify the dynamic obstacle avoidance performance of the algorithm. To ensure the fairness of the simulation experiments, the parameters of the mobile robot, such as the maximum linear velocity, maximum angular velocity, maximum linear acceleration, maximum angular acceleration, linear velocity resolution, angular velocity resolution, time resolution, and prediction period, were compared with those in [15], which were 1 m/s, 20°/s, 0.2 m/s^2, 50°/s^2, 0.01 m/s^2, 1°/s, 0.1 s, and 3 s. The starting point and end point of the path planning of each algorithm are also the same. In the simulation experiment, the map resolution (b) is set to 32, the safety distance (L) is set to 0.2, and the dimension of the safety distance matrix (d_n) calculated according to formula (7) is 11. The initial weight of each evaluation sub-function is set as follows: $\alpha = 0.3$, $\beta = 0.1$, $\gamma = 0.4$, $\eta = 0.2$. The simulation results of the three algorithms are shown in Fig. 10.

As shown in Fig. 10, the red squares are used to simulate dynamic obstacles. As shown in Fig. 10(a), the im-

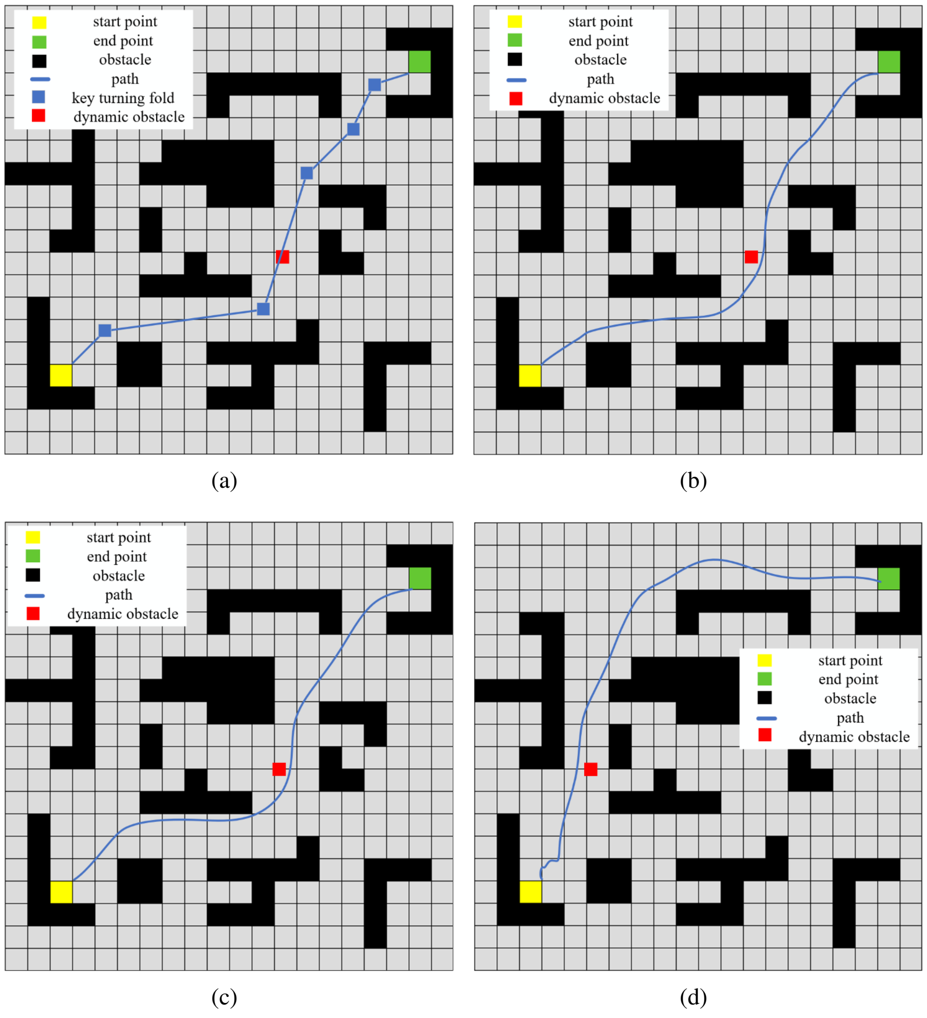


Fig. 10. The simulation results of different fusion algorithms. (a) The path planning of improved A* algorithm. (b) The path planning of fusion method in this paper. (c) The path planning of A*-DWA. (d) The path planning of AC-PAF.

proved A* algorithm cannot avoid dynamic obstacles. As shown in Fig. 10(b), the fusion algorithm in this paper can reach the end point from the starting point on the global path planned by the improved A* algorithm. The three fusion algorithms can reach the end point from the starting point in complex environments, and can avoid dynamic obstacles during the movement. Although the above three fusion algorithms can complete dynamic path planning, comparing Figs. 10(b) and 10(c), it can be seen that the path planned by A*-DWA is relatively close to the obstacle, and the path safety is low. When the robot control accuracy is not high or the environment modeling is inaccurate, the robot can easily collide with the environment. Because the fusion algorithm proposed in this paper adds the safety distance matrix to the A* algorithm, the planned path is far away from the obstacle, and the safety of the path is high. In addition, the path is relatively smooth compared with the improved A* algorithm. The path planned by AC-APF in Fig. 10(d) has redundant path, which is not the optimal path from the starting point to the target point.

In order to further verify the effect of the fusion algorithm, the above three fusion algorithms are quantitatively analyzed with the path length, path search time, and the distance from the path to the nearest obstacle as performance indicators. The results are shown in Table 3.

As shown in Table 3, compared with A*-DWA, the length of the path planned by the fusion algorithm in this paper is reduced by 0.31 m, the minimum distance from the path to the obstacle is increased by 0.05 m, and the search time is reduced by 0.11 s. The reason is that after adding the safety distance matrix to the A* algorithm, the algorithm can sense obstacles around the robot in advance, which improves the path safety and reduces the search time. In addition, after adopting the adaptive weight adjustment strategy, the robot will adjust the weight of the heading angle evaluation sub-function and the obstacle evaluation sub-function in real time according to the positional relationship between itself and the surrounding obstacles, which reduce the path length. The fusion algorithm in this paper is ahead of the AC-PAF in the comparison of each index, which further verifies the superiority. We also simulated the above three fusion algorithms in grid map with the size of 30 × 30 and 40 × 40. The rest of the parameters are the same as above. The results are shown in Figs. 11-12 and Table 4.

As shown in Figs. 11-12 and Table 4, three algorithms can achieve dynamic obstacle avoidance. The performance metrics of the fusion algorithm in this paper are better than the other two algorithms.

Table 3. Comparison result of A*-DWA, AC-APF, and the fusion algorithm of this paper in Fig. 10.

	The path length (m)	The distance to the nearest obstacle (m)	Search time (s)
A*-DWA	5.26	0.02	3.16
AC-APF	6.53	0.05	3.49
This paper	4.95	0.07	3.05

Table 4. Comparison result of A*-DWA, AC-APF, and the fusion algorithm of this paper in scenario 3 and scenario 4.

Scenarios	Algorithm	The path length (m)	DNO (m)	Search time (s)
Scenario 3	A*-DWA	10.35	0.04	5.83
	AC-APF	9.97	0.01	5.26
	This paper	9.46	0.08	4.57
Scenario 4	A*-DWA	15.37	0	7.74
	AC-APF	14.69	0	6.81
	This paper	12.13	0.09	6.25

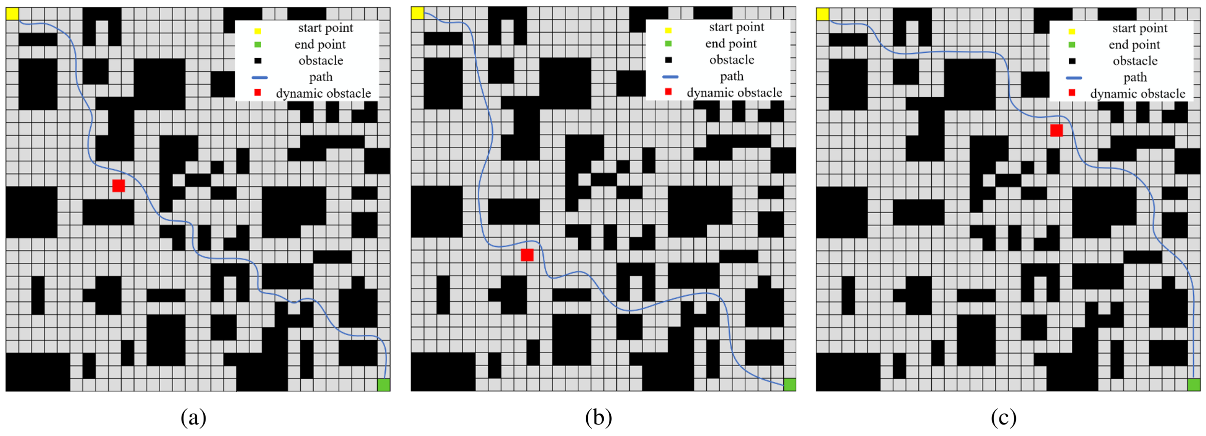
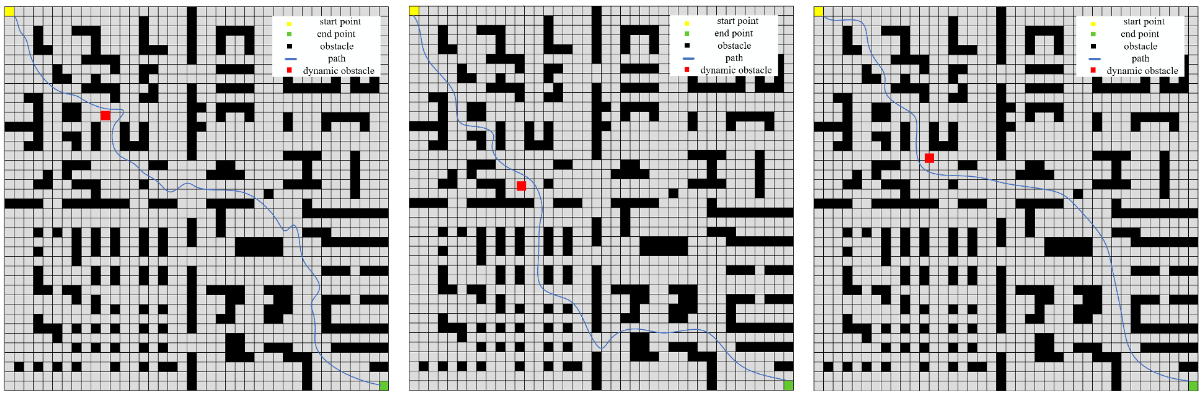


Fig. 11. The simulation results of scenario 3. (a) The path planning of A*-DWA. (b) The path planning of AC-APF. (c) The path planning of fusion algorithm in this paper.



(a) (b) (c)

Fig. 12. The simulation results of scenario 4. (a) The path planning of A*-DWA. (b) The path planning of AC-APF. (c) The path planning of fusion algorithm in this paper.

5.2. Experimental verification of mobile robot based on ROS

The mobile robot is equipped with MPU6050 inertial sensor, lidar, Jetson NX arithmetic unit, and other equipment. The MPU6050 is used to obtain the attitude information of the mobile robot, the lidar is used to obtain the external environment information, and the Jetson NX computing unit runs the ROS operating system. The experimental parameters of the robot are: the maximum linear velocity is 0.6 m/s, the maximum angular velocity is 0.3 rad/s, the number of linear velocity samples is 10, the number of angular velocity samples is 20, and the sampling period is 0.2 s.

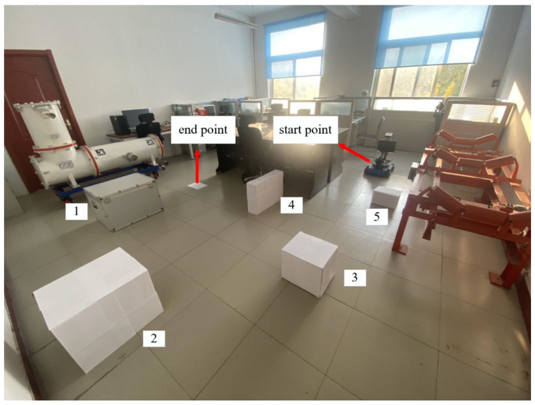
In the experiment 1, no dynamic obstacles were added. The result of experimental 1 is shown in Fig. 13.

As shown in Fig. 13(a), 1-5 represent static known obstacles. As shown in Fig. 13(b), 1-5 represent the map outlines of static known obstacles in Fig. 13(a). The mobile robot can move from the starting point to the end point. The path is farther away from the obstacle. It should be noted that the obstacle was not expanded in this experiment. After adding the safety distance matrix to the A* algorithm, the safety of the path is guaranteed from the algorithm itself. The method proposed in this paper reduces the requirements for the accuracy of the environment map, and provides a new way for the safety of robot operation.

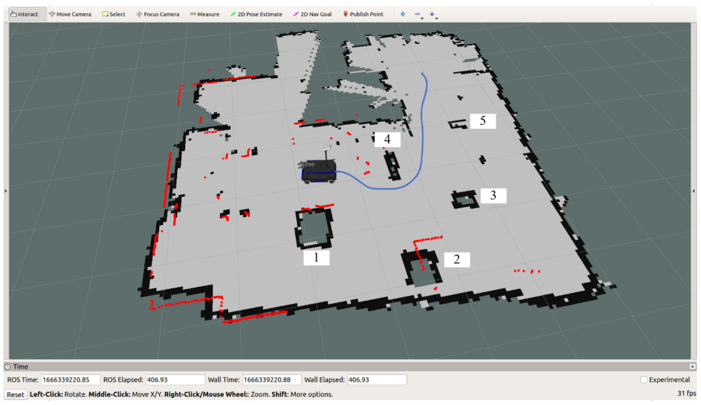
In order to verify the dynamic obstacle avoidance ability of the fusion algorithm, we added a dynamic obstacle to the experimental environment 1. The result is shown in Fig. 14.

As shown in Fig. 14(b), the robot can also run safely from the starting point to the end point after adding the dynamic obstacles. Dynamic obstacles in the experiments are non-mapped obstacles. In addition, it can be seen from Fig. 14(b) that the path planned by the fusion algorithm in this paper has no redundant path, which verifies the effectiveness of the adaptive weight adjustment strategy.

We also did experiment 3 and experiment 4. The results



(a)

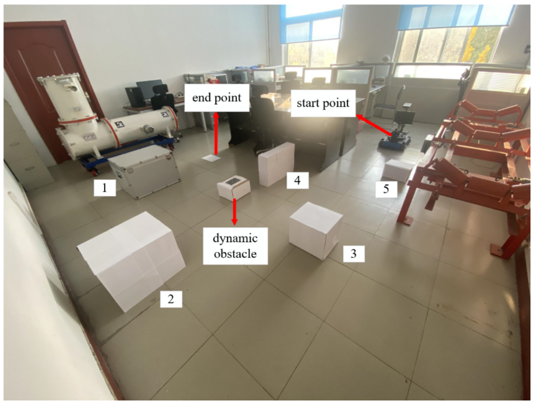


ROS Time: 1666339220.85 ROS Elapsed: 406.93 Wall Time: 1666339220.88 Wall Elapsed: 406.93 Experimental

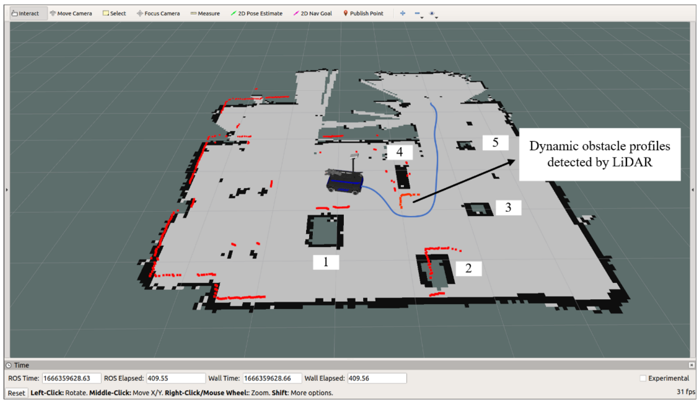
Left-Click: Rotate. Middle-Click: Move X/Y. Right-Click/Mouse Wheel:: Zoom. Shift: More options.

(b)

Fig. 13. The results of experimental 1. (a) Experimental environment 1. (b) The path trajectory of experimental 1.

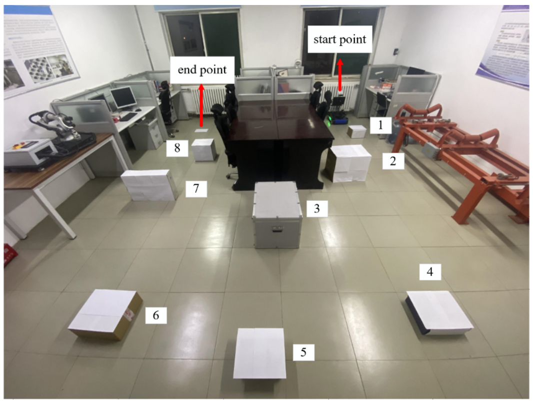


(a)

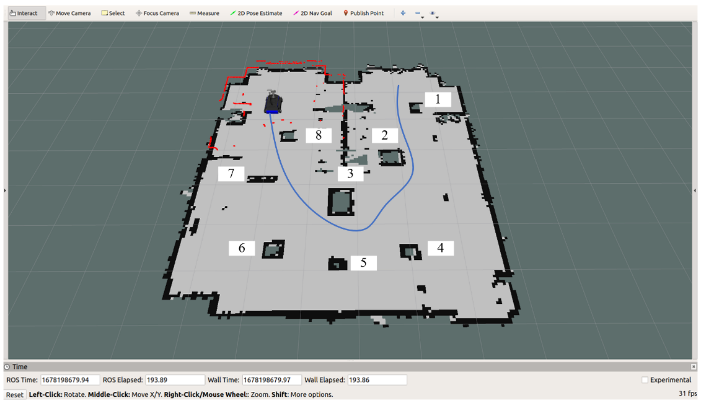


(b)

Fig. 14. The results of experimental 2. (a) Experimental environment 2. (b) The path trajectory of experimental 2.

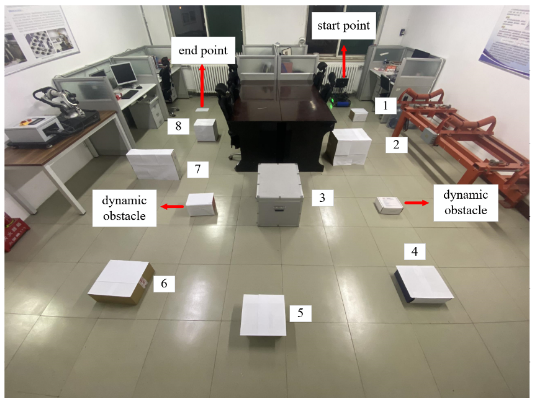


(a)

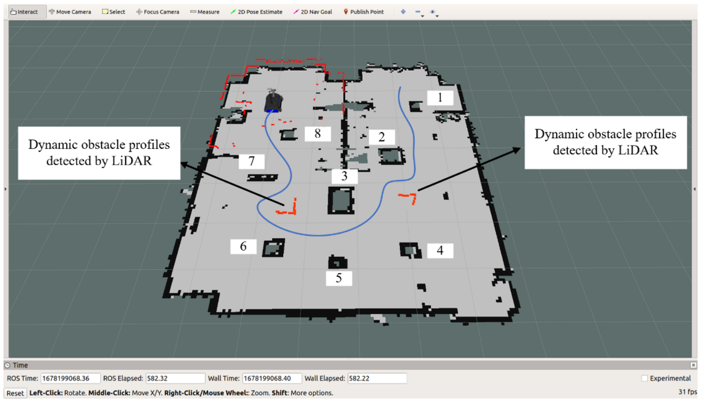


(b)

Fig. 15. The results of experimental 3. (a) Experimental environment 3. (b) The path trajectory of experimental 3.



(a)



(b)

Fig. 16. The results of experimental 4. (a) Experimental environment 4. (b) The path trajectory of experimental 4.

are shown in Figs. 15 and 16.

As shown in Fig. 15(a), there are no dynamic obstacles in experiment 3. As shown in Fig. 15(b), the robot can move safely from the starting point to the end point.

As shown in Fig. 16(a), two dynamic obstacles have been added to environment 3. As shown in Fig. 16(b), the robot will automatically avoid the obstacles and move safely from the starting point to the end point.

6. CONCLUSION

Aiming at the problems that the traditional A* algorithm planned a path with low security, and the DWA algorithm planned a path with redundant length, this paper proposed a path planning algorithm for mobile robots based on a safety distance matrix and an adaptive weight adjustment strategy. By adding the safety distance matrix and the matching heuristic function, the A* algorithm will plan a global path with higher security. The adaptive weight adjustment strategy is introduced into the DWA algorithm, and the robot adjusts the weights of the heading angle evaluation sub-function and the distance evaluation sub-function in real time according to the distance between itself and the obstacle. A new global evaluation function is designed to fuse the above two improved algorithms. In the fusion algorithm, the key fold nodes of the improved A* algorithm are set as the local target points of the DWA algorithm in turn, so that the path planned by the DWA algorithm is closer to the global path. The simulation and experimental results verify that the proposed fusion algorithm in this paper can safely and efficiently complete the path planning task in the actual environment. In the future, the path planning of multi-mobile robots will be studied.

CONFLICT OF INTEREST

The authors declare that there is no competing financial interest or personal relationship that could have appeared to influence the work reported in this paper.

REFERENCES

[1] W. Youn, H. Ko, H. Choi, I. Choi, J. H. Baek, and H. Myung, "Collision-free autonomous navigation of a small UAV using low-cost sensors in GPS-denied environments," *International Journal of Control*, vol. 19, no. 2, pp. 953-968, 2021.

[2] S. O. Park, M. C. Lee, and J. Kim, "Trajectory planning with collision avoidance for redundant robots using Jacobian and artificial potential field-based real-time inverse kinematics," *International Journal of Control, Automation, and System*, vol. 18, no. 8, pp. 2095-2107, 2020.

[3] R. Mao, H. L. Gao, and L. Guo, "Optimal motion planning for differential drive mobile robots based on multiple-interval Chebyshev pseudospectral methods," *Robotica*, vol. 39, no. 3, pp. 391-410, 2021.

[4] J. Song, "Automatic guided vehicle global path planning considering multi-objective optimization and speed control," *Sensors and Materials*, vol. 33, no. 6, pp. 1999-2011, 2021.

[5] T. S. P. H. Pazhohe, R. Akram, and J. S. Mahdi, "Enhanced path planning for automated nanites drug delivery based on reinforcement learning and polymorphic improved ant colony optimization," *Journal of supercomputing*, vol. 77, no. 7, pp. 6714-6733, 2021.

[6] A. S. Nepomniaschaya and M. A. Dvoskina, "A simple implementation of Dijkstra's shortest path algorithm on associative parallel processors," *Fundamenta Informaticae,*, vol. 43, no. 1, pp. 227-243, 2000.

[7] Y. N. Ma, Y. J. Gong, C. F. Xiao, Y. Gao, and J. Zhang, "Path planning for autonomous underwater vehicles: An ant colony algorithm incorporating alarm pheromone," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 1, pp. 141-154, 2019.

[8] D. Lin, B. Shen, Y. R. Liu, F. E. Alassdi, and A. Alsaedi, "Genetic algorithm-based compliant robot path planning: an improved Bi-RRT-based initialization method," *Assembly Automation*, vol. 37, no. 3, pp. 261-270, 2017.

[9] Q. H. Liu, L. J. Zhao, Z. B. Tan, and W. Chen, "Global path planning for autonomous vehicles in off-road environment via an A-star algorithm," *International Journal of Vehicle Autonomous Systems*, vol. 13, no. 4, pp. 330-340, 2017.

[10] Y. P. Lu, X. Yao, and Y. Z. Luo, "Path planning for rolling locomotion of polyhedral tensegrity robots based on Dijkstra algorithm," *Journal of the International Association for Shell and Spatial Structures*, vol. 60, no. 4, pp. 273-286, 2019.

[11] Y. F. Wu, X. X. Zhang, and J. Q. Wu, "Using cellular ant colony algorithm for path-planning of robots," *Applied Mechanics & Materials*, vol. 182, no. 183, pp. 1776-1780, 2012.

[12] J. S. Liu, J. H. Liu, Z. J. Zhang, J. B. Xu, and H. L. Lin, "Anytime RRT based cable automatic routing under three-dimensional environment," *Journal of Mechanical Engineer*, vol. 52, no. 13, pp. 156-165, 2016.

[13] Y. Xin, H. W. Liang, M. B. Du, T. Mei, Z. L. Wang, and R. M. Jiang, "An improved A* algorithm for searching infinite neighbourhoods," *Robot*, vol. 36, no. 5, pp. 627-633, 2014.

[14] J. Chen, L. Xu, J. Chen, and Q. Liu, "Path planning based on improved A* and dynamic window approach for mobile robot," *Computer Integrated Manufacturing Systems*, vol. 28, no. 6, pp. 1650-1658, 2022.

[15] C. Q. Chen, X. Y. Hao, J. S. Li, Z. J. Zhang, and G. P. Sun, "Global dynamic path planning based on fusion of improved A* algorithm and dynamic window approach," *Journal of XI'AN JIAOTONG UNIVERSITY*, vol. 51, no. 11, pp. 137-143, 2017.

[16] H. M. Zhang, M. L. Li, and J. Yang, "Safe path planning of mobile robot based on improved A* algorithm," *Computer Simulation*, vol. 35, no. 4, pp. 324-329, 2018.

[17] X. Cheng and Y. Qi, "Indoor indicator path planning algorithm based on grid method," *Journal of Chinese Inertial Technology*, vol. 26, no. 2, pp. 236-240, 2018.

[18] S. Y. Duan, Q. F. Wang, X. Han, and G. R. Liu, "Improved A-star algorithm for safety insured optimal path with smoothed corner turns," *Journal of Mechanical Engineering*, vol. 56, no. 18, pp. 205-215, 2020.

[19] X. Zhao, Z. Wang, C. K. Huang, and Y. W. Zhao, "Mobile robot path planning based on an improved A* algorithm," *Robot*, vol. 40, no. 6, pp. 904-910, 2018.

[20] X. L. Ma and H. Mei, "Mobile robot global path planning based on improved ant colony system algorithm with potential field," *JOURNAL OF MECHANICAL ENGINEERING*, vol. 57, no. 1, pp. 19-27, 2021.

[21] D. C. Oh and Y. U. Jo, "Classification of hand gestures based on multi-channel EMG by scale average wavelet transform and convolutional neural network," *International Journal of Control, Automation, and System*, vol. 19, no. 3, pp. 1443-1450, 2021.

[22] J. Kim and G. H. Yang, "Improvement of dynamic window approach using reinforcement learning in dynamic environments," *International Journal of Control, Automation, and System*, vol. 20, no. 9, pp. 2983-2992, 2022.

[23] J. Liu, J. Yang, H. Liu, P. Geng, and M. Mao, "Robot global path planning based on ant colony optimization with artificial field," *Transactions of the Chinese Society of Agricultural Machinery*, vol. 46, no. 9, pp. 18-27, 2015.

[24] C. G. Li, X. B. Jiang, W. H. Wang, Q. Cheng, and Y. J. Shen, "A simplified car-following model based on the artificial potential field," *Procedia Engineering*, vol. 137, no. 7, pp. 13-20, 2016.

[25] G. Wang, X. P. Liu, Y. L. Zhao, and S. Han, "Neural network-based adaptive motion control for a mobile robot with unknown longitudinal slipping," *Chinese Journal of Mechanical Engineering*, vol. 32, no. 1, pp. 1-9, 2019.

[26] C. L. Lao, P. Li, and Y. Feng, "Path planning of greenhouse robot based on fusion of improved A* algorithm and dynamic window approach," *Transaction of the Chinese Society for Agricultural Machinery*, vol. 52, no. 1, pp. 14-22, 2021.

[27] J. J. Liu, L. Q. Xue, H. J. Zhang, and Z. P. Liu, "Robot dynamic path planning based on improved A* and DWA algorithm," *Computer Engineering and Applications*, vol. 57, no. 15, pp. 73-81, 2021.

[28] S. Iwamura, Y. Mizukan, T. Endo, and F. Matsuno, "Cable-path optimization method for industrial robot arms," *Robotics and Computer-Integrated Manufacturing*, vol. 73, no. 5, pp. 1-13, 2022.

[29] L. Badilla, D. O. Carrasco, V. F. Sirvent, and H. Villavicencio, "Topological stability for fuzzy expansive maps," *Fuzzy sets and systems*, vol. 425, no. 30, pp. 34-47, 2021.

[30] E. Aguirre and A. Gonzalez, "Integrating fuzzy topological maps and fuzzy geometric maps for behavior-based robot," *International Journal of Intelligent Systems*, vol. 17, no. 3, pp. 333-368, 2012.

[31] Z. L. Gong, Y. H. Gu, T. T. Zhu, and B. Ren, "A method for setting costmap adaptive inflation radius based on robot operating system," *Science Technology and Engineering*, vol. 21, no. 9, pp. 3662-3668, 2021.

[32] J. W. Zhan and Y. Q. Huang, "Path planning of robot combing safety A* algorithm and dynamic window approach," *Computer Engineering*, vol. 48, no. 9, pp. 105-112, 2022.

[33] J. Chen, Z. Y. Zhen, M. Y. Tang, and J. R. Tan, "Robust optimization of uncertain structures based on normalized violation degree of interval constraint," *Computer & Structure*, vol. 182, no. 6, pp. 41-54, 2017.

Xinpeng Zhai received his B.S. degree from School of Electrical Engineering and Automation, Qilu University of Technology, in 2017, and an M.S. degree from Institute of Automation, Shandong Academy of Sciences, in 2020. He is currently pursuing a Ph.D. degree in College of Electrical and Power Engineering, Taiyuan University of Technology. His current research interests include image processing and intelligent robot control.

Jianyan Tian received her B.S. and M.S. degrees in control science and engineering from College of Electrical and Power Engineering, Taiyuan University of Technology, in 1988 and 1993, respectively. She received a Ph.D. degree in system engineering from Nanjing University of Aeronautics and Astronautics, in 2008. She is a Professor at College of Electrical and Power Engineering, Taiyuan University of Technology. Her research interests include modeling of complex systems, intelligent control systems, and intelligent robot.

Jifu Li received his B.S. degree in automation from Beijing Institute of Technology, in 2017, and an M.S. degree in electrical engineering from Texas A&M Univrsity, College Station, in 2019. He is currently working toward a Ph.D. degree in electrical engineering at Taiyuan University of Technology. His current research interests include robotics, machine vision, and GIS partial discharge detection.