applied sciences

MDPI

Article

RRT*-Fuzzy Dynamic Window Approach (RRT*-FDWA) for Collision-Free Path Planning

Lintao Zhou, Nanpeng Wu, Hu Chen *, Qinge Wu and Yingbo Lu

School of Electrical and Information Engineering, Zhengzhou University of Light Industry, Zhengzhou 450002, China
* Correspondence: chenhu@zzuli.edu.cn

Abstract: Path planning is an important aspect and component in the research of mobile-robot-related technologies. Many path planning algorithms are only applicable to static environments, while in practical tasks, the uncertainty in dynamic environments increases the difficulty of path planning and obstacle avoidance compared with static environments. To address this problem, this paper proposes an RRT*-FDWA algorithm. RRT* first generates a global optimal path, and then, when obstacles exist nearby, an FDWA algorithm fixes the local path in real time. Compared with other path planning algorithms, RRT*-FDWA can avoid local minima, rapidly perform path replanning, generate a smooth optimal route, and improve the robot's maneuvering amplitude. In this paper, the effectiveness of the algorithm is verified through experiments in dynamic environments.

Keywords: RRT*-FDWA; maneuver amplitude; dynamic path planning; local minimum

Citation: Zhou, L.; Wu, N.; Chen, H.; Wu, Q.; Lu, Y. RRT*-Fuzzy Dynamic Window Approach (RRT*-FDWA) for Collision-Free Path Planning. *Appl. Sci.* **2023**, *13*, 5234. https://doi.org/10.3390/app13095234

Academic Editor: Jonghoek Kim

Received: 18 March 2023
Revised: 21 April 2023
Accepted: 21 April 2023
Published: 22 April 2023

1. Introduction

In the field of automatic navigation of wheeled mobile robots, path planning is a frequently researched area. The main goal of a path planning algorithm is to plan a collision-free path from the initial state to the target in the constructed space of the robot [1]. Path planning is divided into global path planning and local path planning. Global planning [2–4] is able to accomplish the task in known static environments. However, environments and obstacles change dynamically in most navigation processes. When a mobile robot performs tasks according to global path planning, it must use various devices on board, such as lidar, depth cameras, and IMUs, to sense the surrounding environment and its own state and quickly plan local collision-free paths. For navigation, robots therefore usually use a combination of global path planning and local path planning [5–7].

The Rapid Exploration Randomized Tree [8] (RRT) has been shown to be a very effective global path planning algorithm; RRT can plan paths quickly and simply by generating a tree structure at high-speed increments in the construction space to find a goal. However, RRT algorithms also possess some constraints, such as not having asymptotic optimality [9,10] and an the inability to avoid dynamic obstacles; variants of improved RRT algorithms have, therefore, been proposed. Jin-Gukang [11] et al. proposed a "Post-Triangular Rewiring" method, which reduces the path planning time, improves efficiency, and creates an algorithm close to the global optimum. Moon C [12] et al. proposed a dual-tree fast exploration random tree (DT-RRT) algorithm to decrease the computational complexity of the RRT. In addition, DT-RRT reduces the length of the path and increases the coverage of the sampling points. Nasir J et al. proposed an RRT* [13] algorithm which can make the path asymptotically optimal. Chen L et al. [14] introduced the dual-tree structure into an RRT* algorithm, thereby creating separate extension and optimization processes and improving the convergence speed. In addition, some scholars have attempted to improve global path planning methods such as RRT* to avoid dynamic obstacles. Qi et al. [15] applied the RRT* algorithm to dynamic environments and introduced Pareto

theory to propose a multi-objective dynamic fast exploration stochastic algorithm (MOD-RRT*), which improved the effective performance of the RRT* algorithm in avoiding unknown obstacles.

However, there are many limitations in global path planning and dynamic environments. Compared with static environments, there are many uncertainties, such as randomly moving obstacles. It is, therefore, very difficult to predict the movement path of obstacles. Thus, local path planning in dynamic environments is currently an area of great research interest.

In recent years, researchers have proposed many local path planning algorithms, such as the dynamic window approach [16] (DWA) and the artificial potential field method [17] (APF). Furthermore, Young-In Choi [18] et al. proposed a collision avoidance algorithm based on the D*Lite algorithm for mobile robots; this algorithm was applied to a logistics delivery scenario to effectively avoid collision and safely reset the robot's optimal movement route when the robot encounters a crossover situation at an intersection. Kumar [19] et al. proposed a combination of the sine cosine algorithm and ant colony algorithm for multiple environments such as static and dynamic, and applied the algorithm in multi-robot formations. Guo [20] et al. constructed a risk region of dynamic obstacles using the Kalman filter state estimation, combining it with a nonlinear model predictive control to achieve safe obstacle avoidance. DWA has been extensively applied because of its dynamic characteristics combined with those of the robot. Later, Zhong [21] et al. proposed an adaptive rolling window method based on the edges of obstacles and the target point, which has good safety and environmental applicability. Chang [22] et al. combined Q learning with the DWA algorithm and proposed two new evaluation functions. This demonstrated a high navigation efficiency and success rates in complex unknown environments, but incurred a higher time cost. Xiang [23] et al. implemented adaptive weight coefficients for the DWA algorithm for complex environments, making the path of the mobile robot smoother when avoiding obstacles. Many new and improved algorithms have been derived for obstacle avoidance in dynamic environments. Wu [24] et al. combined the A* algorithm and the DWA algorithm to produce an algorithm that was closer to the global optimal path but with less smoothness.

Nevertheless, the improved DWA algorithms are still unable to avoid the problem of local optimal solutions and may fail to complete the task due to the lack of global planning. An algorithm may get stuck at local minima during planning and consume more energy. A suitable path planning algorithm should be able to plan a complete collision-free path that satisfies the robot dynamics.

Therefore, to resolve the path planning problem in a dynamic environment, this paper proposes an RRT*-FDWA algorithm. The contribution of the proposed algorithm to the path planning problem is as follows:

- A fuzzy controller is added to the adaptive weight index of the DWA algorithm. This makes the weights adaptive, improves the safety of path planning, and enables timely avoidance of dynamic obstacles.
- By combining the RRT* algorithm with the FDWA algorithm, local planning can avoid hitting local minima. The RRT*-based path re-planning is able to replan the global path after local path planning; this enhances the robot's maneuverability and improves target fit.

The manuscript is divided into the following parts: Section 2 explains the original algorithm, the dynamics required to move the robot, and the related work. Section 3 discusses the FDWA algorithm and the fuzzy control principle. Section 4 discusses the RRT*-FDWA algorithm. Section 5 provides experimental results to demonstrate the effectiveness of the RRT*-FDWA. Section 6 discusses the results and future work.

2. Related Work

Collision-free path planning is usually divided into two stages. The first stage is global path planning—RRT* in this paper. The second stage is local path planning, for which this paper adopts the FDWA algorithm.

2.1. RRT and DWA Algorithm*

The RRT* algorithm is mainly improved by the RRT algorithm. The RRT* algorithm adds *Rewire*, by routing the algorithm to reselect the parent node, thereby achieving the global optimal. The RRT* algorithm, therefore, has probabilistic completeness and asymptotic optimality.

Path planning in the DWA first calculates the current sampled velocity range based on the mobile robot's own characteristics. The sampled acceleration and angular velocity are used to simulate the trajectory of the robot in a certain range, and the trajectory in the range is evaluated using an evaluation function with certain rules; after the trajectory in the range has been obtained, the optimal path is selected to enable the robot to move.

2.2. Kinematic Model

Mobile robots can be divided into omnidirectional and non-omnidirectional robots according to the kinematics of casters, and the robot used in this paper is a non-omnidirectional mobile robot. In Figure 1, XOY represents the global coordinate system. x_1Py_1 is the local coordinate system, x_1 is the linear movement direction of the robot, y_1 is perpendicular to the x_1 axis, P is the center of the robot as the origin of the local coordinate system, and θ is the heading angle. Thus, the robot's global coordinates, position ε, can be obtained, as shown in Equation (1). To map the global coordinate system to the local coordinate system, the rotation matrix R_θ is used. Equation (3) is converted from global coordinates to local coordinates $R\varepsilon$.

$$\varepsilon = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} \tag{1}$$

$$R_\theta = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2}$$

$$R_\varepsilon = R_\theta \times \varepsilon \tag{3}$$

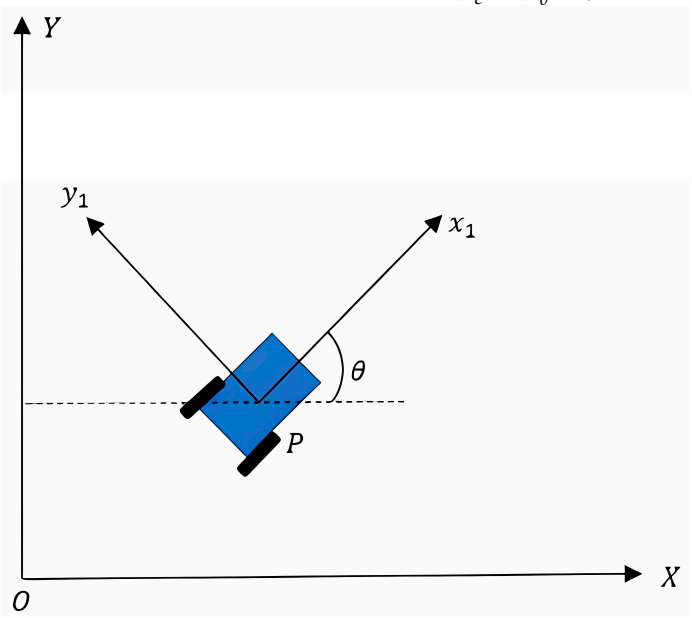


Figure 1. Reference system coordinates.

In a non-omnidirectional moving robot, the robot only has a linear velocity V in the x_1 direction and an angular velocity ω of rotation. Suppose the sampling time is ΔT, the distance the robot travels is ΔS, and the coordinates of the robot are $\Delta X, \Delta Y$.

$$\Delta S = V * \Delta T \tag{4}$$

$$\begin{cases} \Delta X = V * \Delta T \cos\theta_T \\ \Delta Y = V * \Delta T \sin\theta_T \end{cases} \tag{5}$$

According to the coordinates of the robot, $X_{n,}Y_n$, the current sampling time can be obtained by summing up its state and increment.

$$\begin{cases} X_n = X_{n-1} + V * \Delta T \cos\theta_T \\ Y_n = Y_{n-1} + V * \Delta T \sin\theta_T \end{cases} \tag{6}$$

The DWA algorithm [25–27], in Equation (6), has three evaluation metrics. They are the azimuth evaluation function, $heading(v, w)$, which denotes the angular difference between the robot and the target point. $dist(v, w)$, which denotes the distance between the robot and the nearest detected obstacle, is the distance function. $velocity(v, w)$ denotes the relative velocity magnitude of the robot trajectory. α, β, and γ denote the weight coefficients of the function and σ denotes the normalization required to obtain the evaluation function $G(v, w)$.

$$\gamma = 1 - \beta - \alpha \tag{7}$$

$$G(v, \omega) = \sigma(\alpha * heading(v, \omega) + \beta * dist(v, \omega) + \gamma * velocity(v, \omega)) \tag{8}$$

3. FDWA Algorithm

The RRT* algorithm is able to perform global path planning in an already established map model environment or with static obstacles. However, in the real environment, there are many uncertainties, such as unknown obstacles. If global path planning alone is used, the robot will easily collide with obstacles in such unknown environments. Therefore, in order to achieve the dynamic obstacle avoidance capability of a mobile robot, this paper incorporates a local path planning algorithm—the FDWA—for local dynamic obstacle avoidance.

3.1. FDWA Fuzzy Distribution

In the DWA algorithm, the percentage of weights has a very strong influence on the results of the path planning algorithm. Therefore, in order to improve the efficiency of the algorithm and be better able to adapt to the complex environment, the fuzzy inference and DWA algorithm are combined, enabling adaption of the evaluation index weights. Fuzzy inference is performed on the azimuth evaluation function weights α and distance function weights β, (β, $\alpha \in [0, 0.5]$), and a dual-input, dual-output fuzzy controller is designed. The angle θd at which the actual trajectory of the robot deviates from the predetermined trajectory and the distance Od to the nearest obstacle are used as dual inputs, and the azimuth evaluation function weight, α, and the distance function weight, β, are used as dual outputs. The membership function of the fuzzy system uses trigonometric functions, which are suitable for describing well-defined ranges and information precisely. The input membership functions θd and θd are shown in Equation (10): ($\mu(u) \in [0, 1]$]). The output affiliation function of weight α is $\mu(o)$, shown in Equation (11): ($\mu(o) \in [0, 1]$). Figure 2a shows the input membership function and Figure 2b shows the output.

$$\mu(u) \in [O_d, \theta_d], \mu(o) \in [\beta, \alpha] \tag{9}$$

$$\mu(u) = \begin{cases} 1 - 2u & 0 \le u \le 0.25 \\ 2u & 0.25 < u < 0.5 \\ 2 - 2u & 0.5 \le u \le 0.75 \\ 2u - 1 & 0.75 < u < 1 \end{cases} \tag{10}$$

$$\mu(o) = \begin{cases} 8o & 0 \le o \le 0.125 \\ 1 - 8o & 0.125 \le o \le 0.1875 \\ 8o - 1 & 0.1875 \le o \le 0.25 \\ 3 - 8o & 0.25 \le o \le 0.2975 \\ 8o - 2 & 0.2975 \le o \le 0.375 \\ 4 - 8o & 0.375 \le o \le 0.5 \end{cases} \tag{11}$$

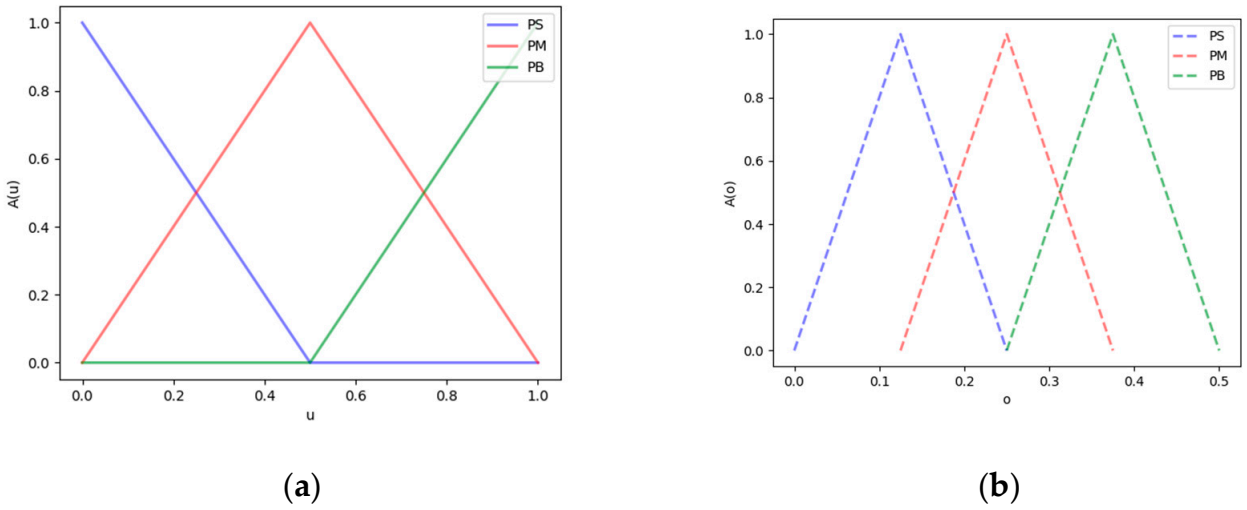


Figure 2. Membership function graph. (**a**) Input membership; (**b**) output membership.

3.2. Fuzzy Rules and Clarification

First, the input and output variable domains of the fuzzy controller are defined as [0, 1], which is a continuous domain. The fuzzy sets are defined as 0, PS, PM, and PB, i.e., zero, positive small, positive medium, and positive large. Tables 1 and 2 show the design method fuzzy rules, α and β, as follows.

1. When O_d, θ_d, $\in$ PB. The mobile robot should reduce the value of θ_d,θ_d as shown in Figure 3, $\theta_d = |\theta_1 - \theta_2|$, and should reduce the value of α and β so that the difference between the current trajectory and the desired trajectory is reduced and the path is smoother. This will make, the velocity function increase in weight and make the robot accelerate toward the target.
2. When $\theta_d \in PB, O_d \in PS$. The robot will reduce the value of α to make the path smoother. The β value should be large to make the robot avoid the obstacle.
3. When $\theta_d \in PS$, $O_d \in PB$. The value of α should be made moderate to maintain the smoothness. The value of β should be deceased; this will make the velocity function increase its share in the weight and make the robot accelerate toward the target.
4. When θ_d, $O_d \in PS$. The value of α should be increased and the value of β should be increased to ensure that the robot passes the obstacle safely.

Table 1. α rule list.

θ_d \ O_d	0	PS	PM	PB
0	PB	PB	PS	0
PS	PB	PB	PS	0
PM	PB	PM	PM	0
PB	PB	PM	PM	PS

Table 2. β rule list.

θ_d \ O_d	0	PS	PM	PB
0	PB	PB	PM	PS
PS	PB	PB	PM	PS
PM	PB	PB	PM	0
PB	PB	PB	PS	0

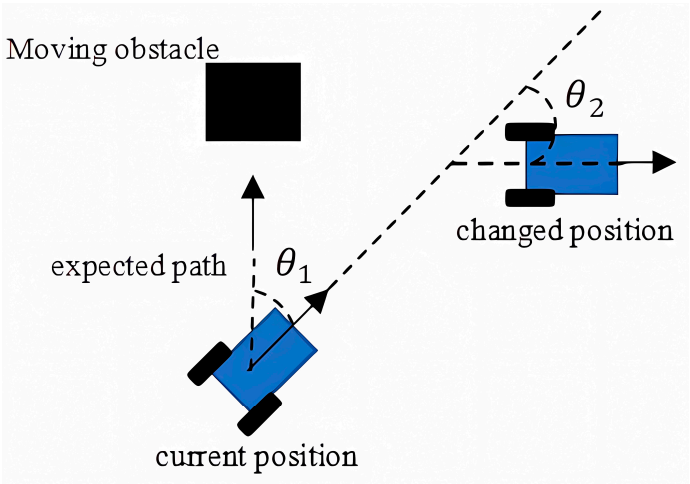


Figure 3. Change in heading angle.

The fuzzy logic inference uses the Mamdani inference method. The clarity value U is (α, β). The exact value of α, β at the current moment is obtained after defuzzification to prevent the FDWA algorithm from falling into local minima. The maximum membership averaging method can effectively eliminate extreme cases, so the maximum membership averaging method is used to solve the problem, as in Equation (12). $A(o_j) = \max(A(o))$, $j = 1, 2 \ldots n$.

$$U = \sum_{j=1}^{n} A(o_j)/2 \tag{12}$$

4. Dynamic Path Planning Algorithm Based on RRT*-FDWA

4.1. RRT-FDWA Algorithm Flow*

The RRT*-FDWA path planning algorithm uses both the RRT* and FDWA algorithms. When the global optimal path is completed according to the RRT* algorithm, a path replanning judgment is performed. When a moving obstacle is encountered, the FDWA algorithm is used for local path planning; after successfully avoiding the obstacle, the RRT* algorithm plans a new global path for the mobile robot to continue driving. The RRT*-FDWA process algorithm is shown in Figure 4.

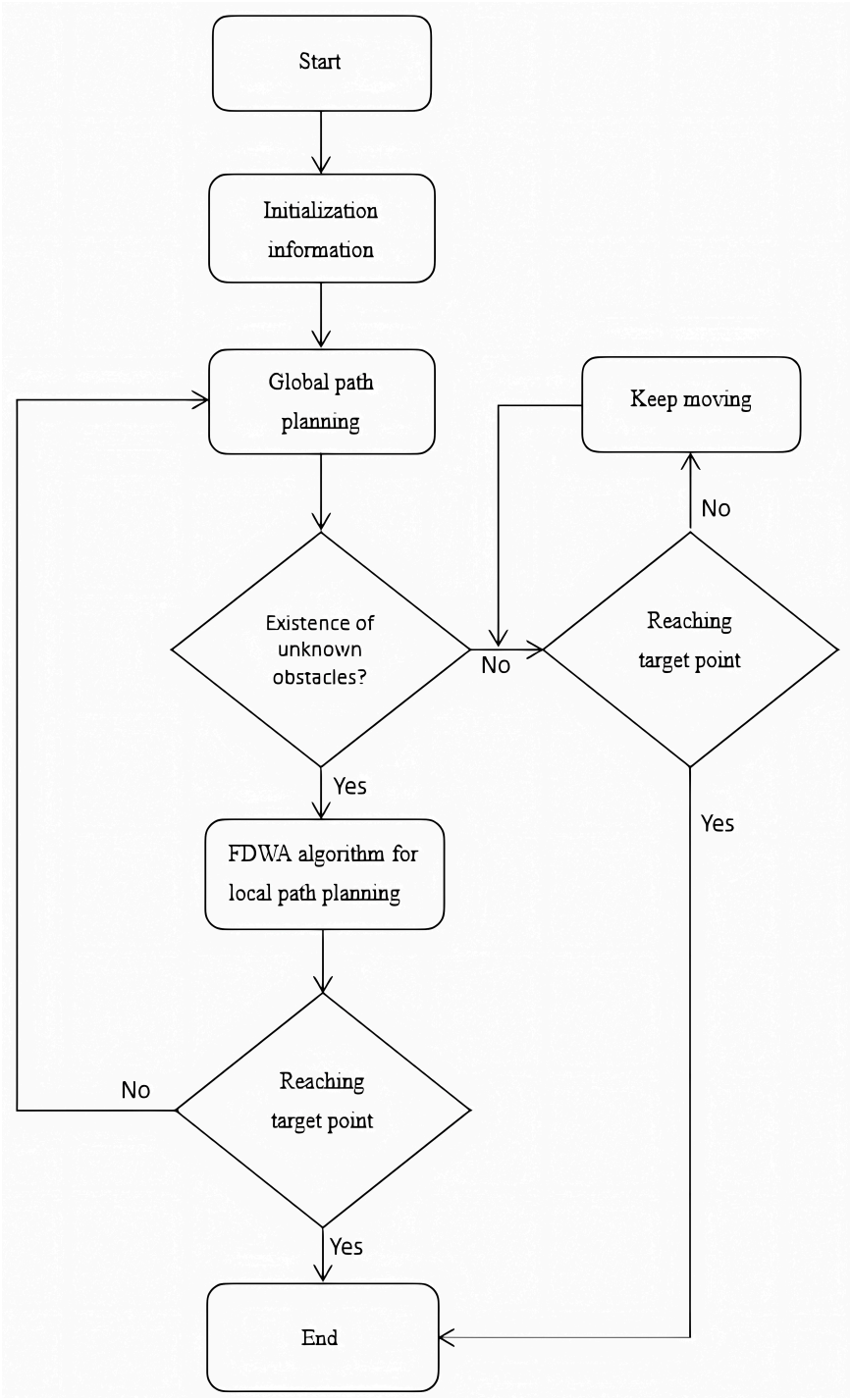


Figure 4. RRT*-FDWA flow chart.

The algorithm flows as follows:

Step 1: Create a map of the known environment and set target points.

Step 2: The RRT* algorithm plans a collision-free global optimal path based on the already established map environment and its own sensors, such as lidar.

Step 3: Determine whether the robot can reach the target point according to the end condition of the algorithm; if so, end the algorithm; if not, continue the execution.

Step 4: Determine whether there is an environment or moving obstacle in the established map; if the lidar determines that the moving obstacle is dangerous, jump to the FDWA algorithm for local path planning to avoid the obstacle.

Step 5: Perform a new global path plan, determine the optimal path, and continue driving along the global path. Judge whether it is possible to reach the target point: if 'No', jump to the Step 3 loop; if 'Yes', end the algorithm. Figure 5 represents the global path planning, encountering moving obstacles in the path replanning process.

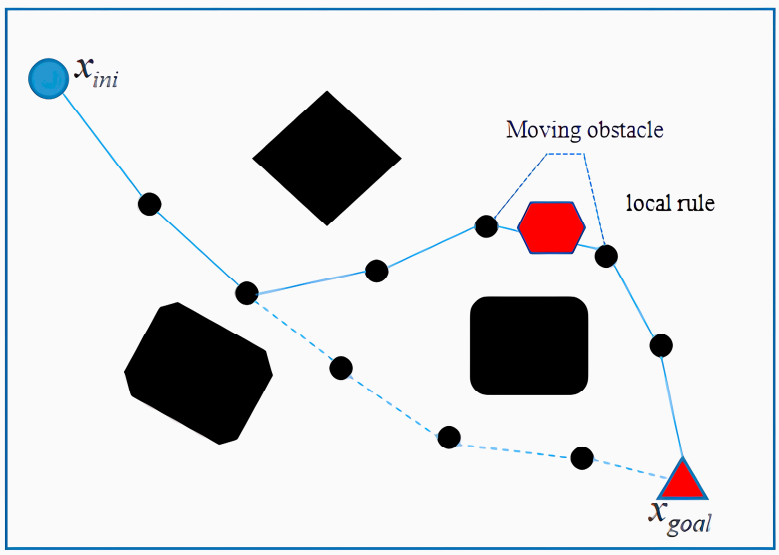


Figure 5. Path planning diagram.

4.2. RRT-FDWA Local Minima and Pseudocode*

In the DWA algorithm, the overall generation value increases as the value of its weight coefficient increases the closer it gets to the obstacle. This will cause the robot to fall into local minima during local path planning. In the RRT*-FDWA algorithm, to avoid local minima, the robot can quickly enter global path planning once it has successfully avoided an obstacle, as shown in Equation (13). K is the reward and punishment function of the DWA algorithm.

$$G'(v,\omega) = K \cdot G(v,\omega) \tag{13}$$

$$\Delta t(v,\omega) = \Delta S/\Delta V \tag{14}$$

$$K = \begin{cases} a & t < 0.5 \\ 1 & t = 0.5 \\ b & 0.5 < t < 1 \\ 0 & t > 1 \end{cases} \tag{15}$$

In Equation (14), $\Delta t(v,\omega)$ is the time function with the obstacle. In Equation (15), a and b represent the reward and punishment function K, where $a < 1 < b$. When the value of t is less than 0.5, it is not caught in the local minimum; when t is greater than 0.5, it is regarded that the robot is caught in the local minimum. Global path planning uses the RRT* [25] algorithm, while local path planning uses the FDWA algorithm. The pseudo code of the RRT*-FDWA algorithm is shown in Algorithm 1; Algorithm 2 is the *Rewire* in Algorithm 1.

Algorithm 1 RRT*-FDWA Algorithm.

input: p_{ini}, p_{goal}, Map, v, w, K
output: $p_{ini \to} p_{goal}$ $path$
WHILE Target_not_reach DO
FOR i in range(n): $p_{rand} \leftarrow Sample(Map$
$p_{nearest} \leftarrow Node_{list}(p_{rand}, p_{ini})$
$p_{new} \leftarrow Steer(p_{nearest}, p_{rand})$
$Cost(x_{new}) \leftarrow d(p_{nerarest}, p_{new}) + Cost(p_{nearest})$
IF (*Check_collision*($p_{nearest}, p_{new}$)) *THEN*
$p_{near} \leftarrow findnear_{nodes(p_{new})}$
$p_{new} \leftarrow C\_parent(p_{near}, p_{new})$
$Node\_list \leftarrow append(p_{new})$
$p_{parent} \leftarrow Rewire(p_{new}, p_{near})$
IF (*Check_ Moving obstacle* ($p_{nearest}, p_{new}$)) *THEN*
$DWA(p_{near}, p_{new}) \leftarrow G'(v, w), K$
$G'(v, w), K \leftarrow Rule(\alpha, \beta)$
$\alpha, \beta \leftarrow clarification$
IF (*Check_collision*($p_{nearest}, p_{new}$)) *THEN*
end
$IF\left(p_{new} \to p_{goal}\right) THEN$
$IF(Check\_conlision(p_{new}, p_{goal}))$ *THEN*
$Temppath \leftarrow cunrrrent\left(p_{ini} \to p_{goal}\right)$
Return path
End while

Algorithm 2 Rewire (P_1, P_2).

IF Collision_free (p_1, p_2)*THEN*
$p_{parent2} \leftarrow p_2[i-1]$
$IFCost(p_2) > Cost(p_1))$ *THEN*
$IF(Check\_collision(p_{nearest}, p_2))THEN$
$p_{parent} \leftarrow p_2$
$Cost \leftarrow Cost(p_2)$
End if
End if

In Algorithms 1 and 2 in this paper, the following terms are used:

Sample (Map): The number of sampling points generated randomly on the map towards the target node is in the range [0, 100].

Node_list (p_1, p_2): The set of Euclidean distances between the sampling point and the parent node, where the coordinates of the random sampling point and the parent node are p_s (x_1, y_2) and p_p (x_2, y_2), respectively.

$p_{nearest}$: Minimum of *Node_list* (p_1, p_2) min (*Node_list*)

Steer (p_1, p_2): The angle between p_1 and the line connecting p_1 and p_2 of its parent node; multiply the steering angle by the step size β to get p_{new}. Let the steering angle be θ; p_{new} is obtained by the following Equations (16) and (17).

$$p_{new,x} = \beta \cos\theta \tag{16}$$

$$p_{new,y} = \beta \sin\theta \tag{17}$$

Check_collision (p_1, p_2): Check if there are obstacles between p_1 and p_2.

Target_not_reach: If or not the target point is reached: the return value is 1 if the target point is reached and 0 if the opposite is true.

Cost(x): Return along the total path length of the target node from node p.

find near_nodes (p_1): Find a new parent node near node p again at random.

C_parent (p_1, p_2): Add a new optional parent node and define its length, angle, and generation value.

D (p_1, p_2): Denotes the Euclidean distance between nodes p_1 and p_2.

Rewire (p_1, p_2): Determine whether p_2 can replace p_1 as the new parent node.

Collision_free (p_1, p_2): Check whether it is feasible and that there are no obstacles between node p_1 and node p_2.

G' (v, ω): Evaluation function of the DWA algorithm with local minimum judgment added.

Rule (α, β): Fuzzy rule table for α and β.

5. Experimental Results

In this paper, the algorithm was simulated in an AMD Ryzen 7 5800H 3.2 GHz and 16 G RAM computer, and a Matlab simulation was used (Matlab version 2019b). A two-wheel differential speed mobile robot with a universal wheel-manipulable wheel in the front was used.

5.1. RRT*-FDWA Global Path Planning

First, RRT* was used to simulate the global planning path. As shown in Figure 6, for the random moving obstacle environment, dashed and solid lines represent the global path. The dashed line shows the process of the RRT* algorithm rewiring and selecting a new node for the first time; the blue circle represents p_{new} and the black objects represent the moving obstacles. A size [10,10] map with a start point of [1.2,1.2] and a target point of [9,9] was used.

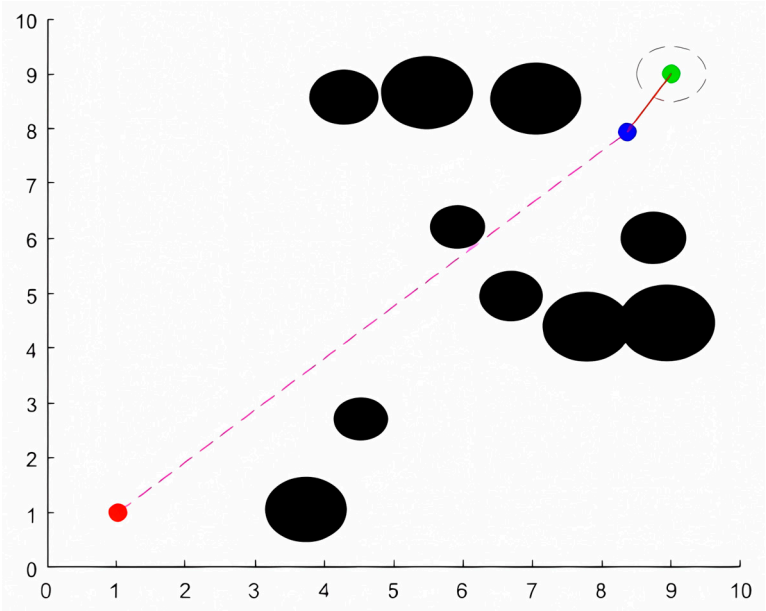


Figure 6. RRT*-FDWA global path planning.

5.2. FDWA Lobal Path Planning

In order to verify its effectiveness, the FDWA algorithm was simulated. First, the initial values of the weight coefficients in the evaluation function were set at $\alpha = 0.2$, $\beta = 0.4$, and $\gamma = 0.4$; the maximum linear and angular velocities were also set to $v = 0.5$ m/s and $\omega = 0.3$ rad/s. a and b represent the reward and punishment function K: $a = 0.5$ and $b = 1.5$. The time for forward simulation was 3 s and dt = 0.1.

Figure 7 shows the FDWA algorithm planning diagram and demonstrates the effectiveness of the algorithm in the presence of unknown obstacles. The solid purple line indicates the path simulated by the FDWA algorithm in the constant forward direction, and the dashed line indicates the final completed trajectory. The blue solid circles indicate

stationary obstacles; the blue solid hollow circles indicate the current position of the moving obstacle; and the blue dashed circles indicate the position of the moving obstacle at the previous moment. When the mobile robot simulates the path forward, an optimal local path is found according to the evaluation function and the local minimum problem can be successfully avoided.

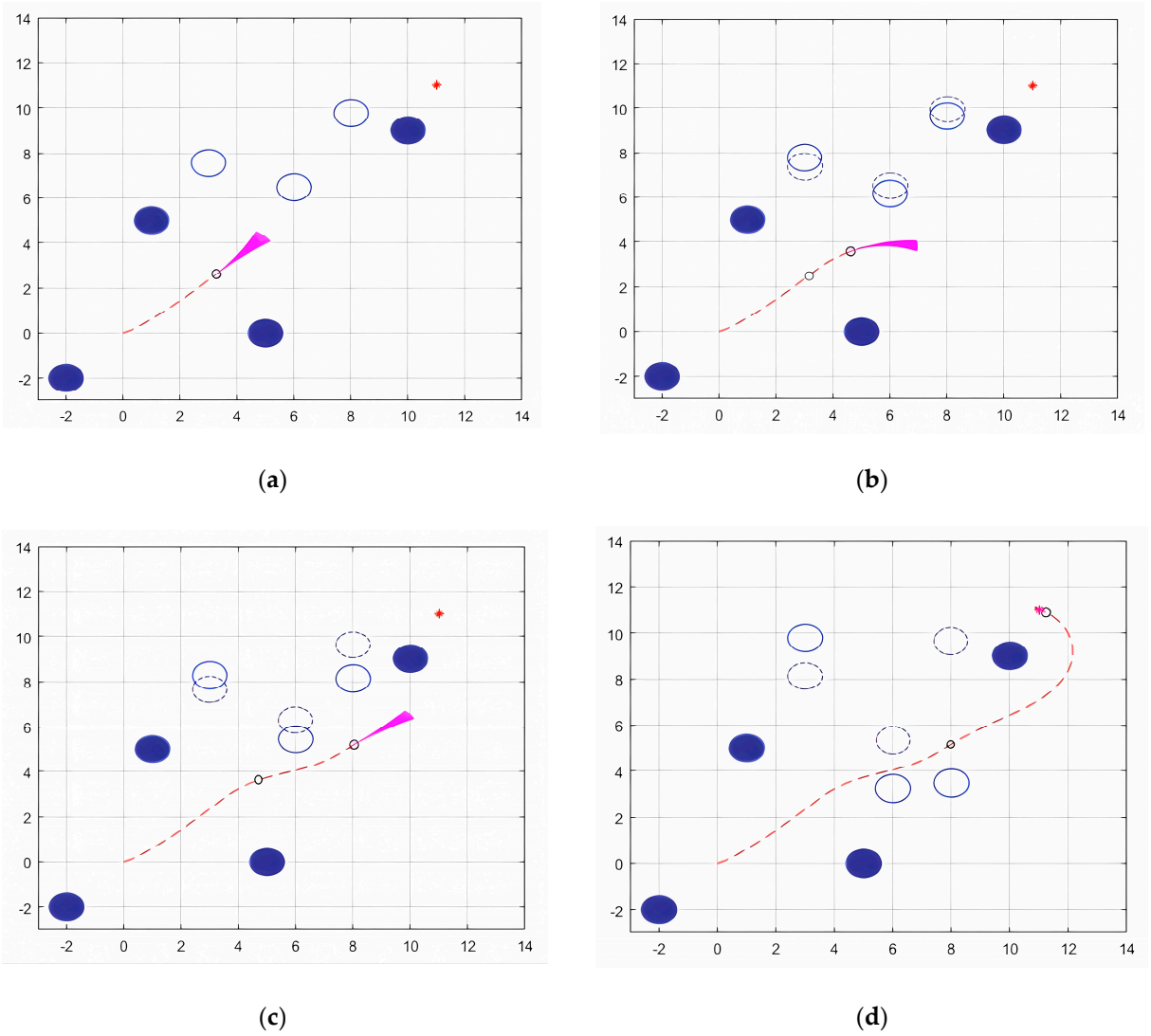


Figure 7. FDWA path planning diagram. (**a**) Position planning diagram at the moment t = 0.5. (**b**) Position planning diagram at the moment t = 1. (**c**) Position planning diagram at the moment t = 3. (**d**) Position planning diagram at the moment t = 5.

To further verify the effectiveness of the FDWA, this paper is validated by a real robot. Figure 8a shows the experimental robot tianbot_mini, a two-wheel differential drive robot with a drive wheel at the rear and a gimbal at the front, and with classic PD control for the drive wheel. It has a ydlidar x2 LIDAR for obstacle detection. The linear velocity is constrained, $v \in (0, 0.5\ \mathrm{m/s})$, and the FDWA algorithm is encapsulated as a local path planner for the ROS navigation package. The experimental environment is the same as the simulation environment, and the white mobile robot in Figure 8b is regarded as a moving obstacle with a handle control and a speed of 0.10 m/s. Figure 8b–f shows the FDWA path planning process, and the experimental results better validate the effectiveness of the algorithm.

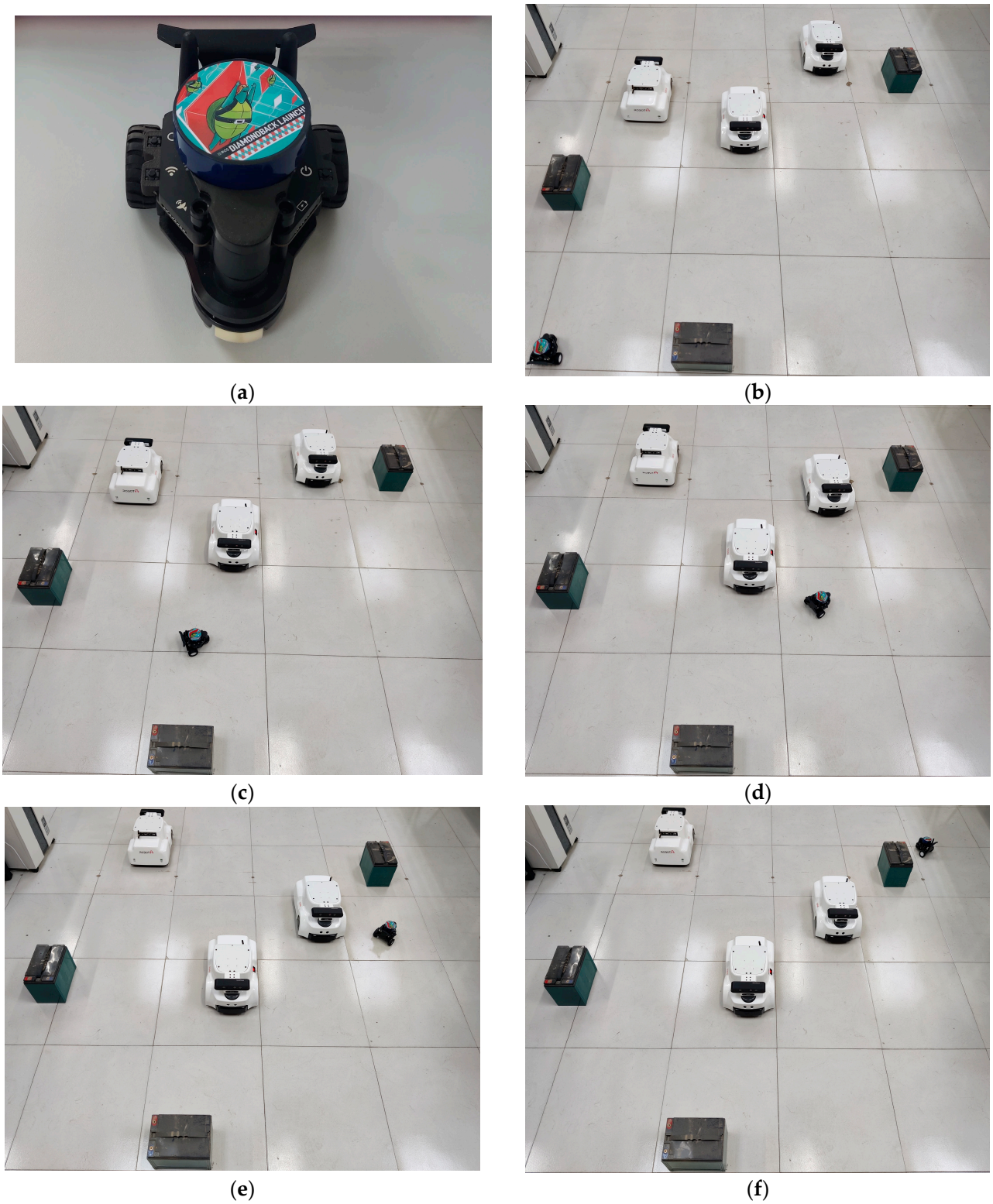


Figure 8. FDWA real environment path planning map. (**a**) the experimental robot (**b**) The robot position map at the moment t = 0. (**c**) The robot position map at the moment t = 3. (**d**) The robot position map at the moment t = 5. (**e**) The robot position map at the moment t = 8. (**f**) The robot position map at the moment t = 10.

5.3. Experimental Comparison

In this paper, we used a two-wheel differential speed robot for the simulation. The maneuverability degree, δ_m, of this mobile robot can, therefore, be obtained as two, and according to Equation (18), $rank[C_\alpha(\beta_\alpha)]$ is the number of maneuverable wheels.

$$\delta_m = 3 - rank[C_a(\beta_a)] \tag{18}$$

In dynamic obstacle avoidance, robot behavior includes deceleration or acceleration to avoid obstacles when it encounters them, but such behaviors may cause damage to safety. In this paper, the behavior of avoiding obstacles is proposed as the standard deviation formula for speed, the smaller value of which represents higher safety, as shown in Equation (19), where v_i indicates the line speed at the current moment. The integrated speed difference formula presents the maneuvering magnitude, which indicates the robot's evasion ability when encountering obstacles, and the higher its value, the better the evasion ability, as shown by Equation (20).

$$\sigma_w^2 = \sqrt{\frac{\sum_{i=1}^{n}(v_i - v)^2}{n}} \tag{19}$$

$$\delta_r = \sigma_w^2 \delta_m / \theta_d \times 100\% \tag{20}$$

Figure 9 shows the RRT*-FDWA path planning process; the solid red circles indicate the current position and solid blue circles indicate p_{new}. Hollow circles indicate the previous moment position and the planned route and dashed hollow circles indicate the moving obstacle at the previous moment.

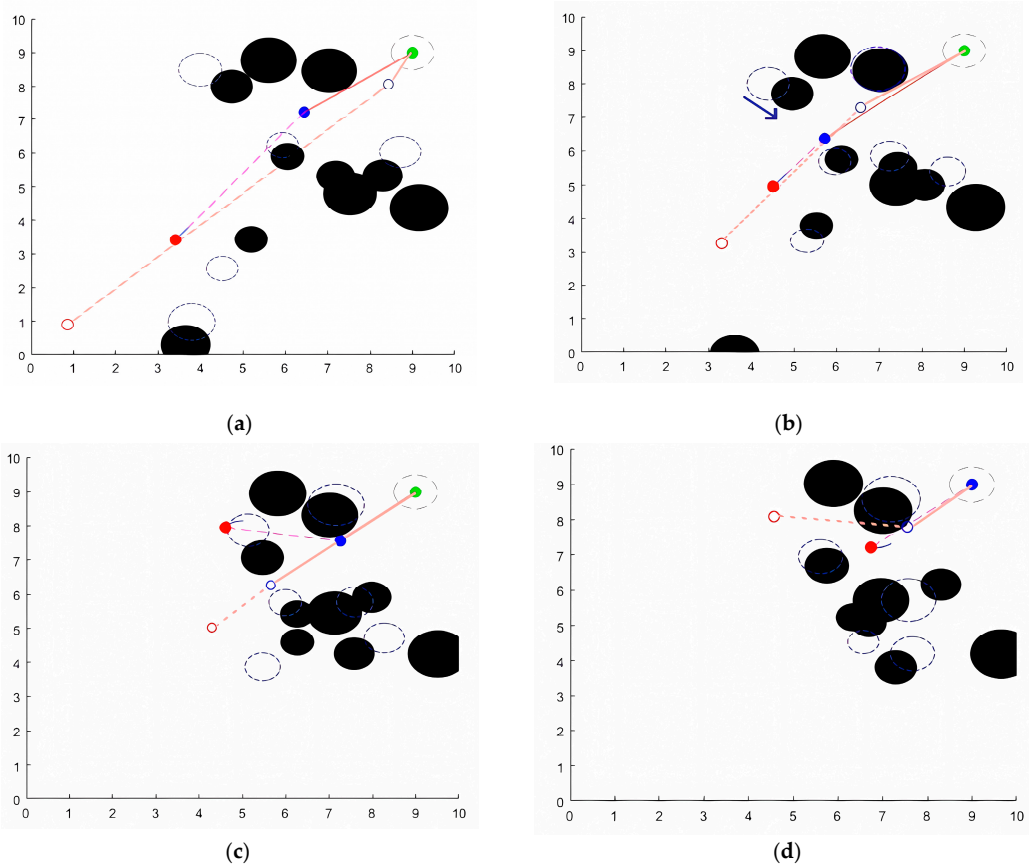
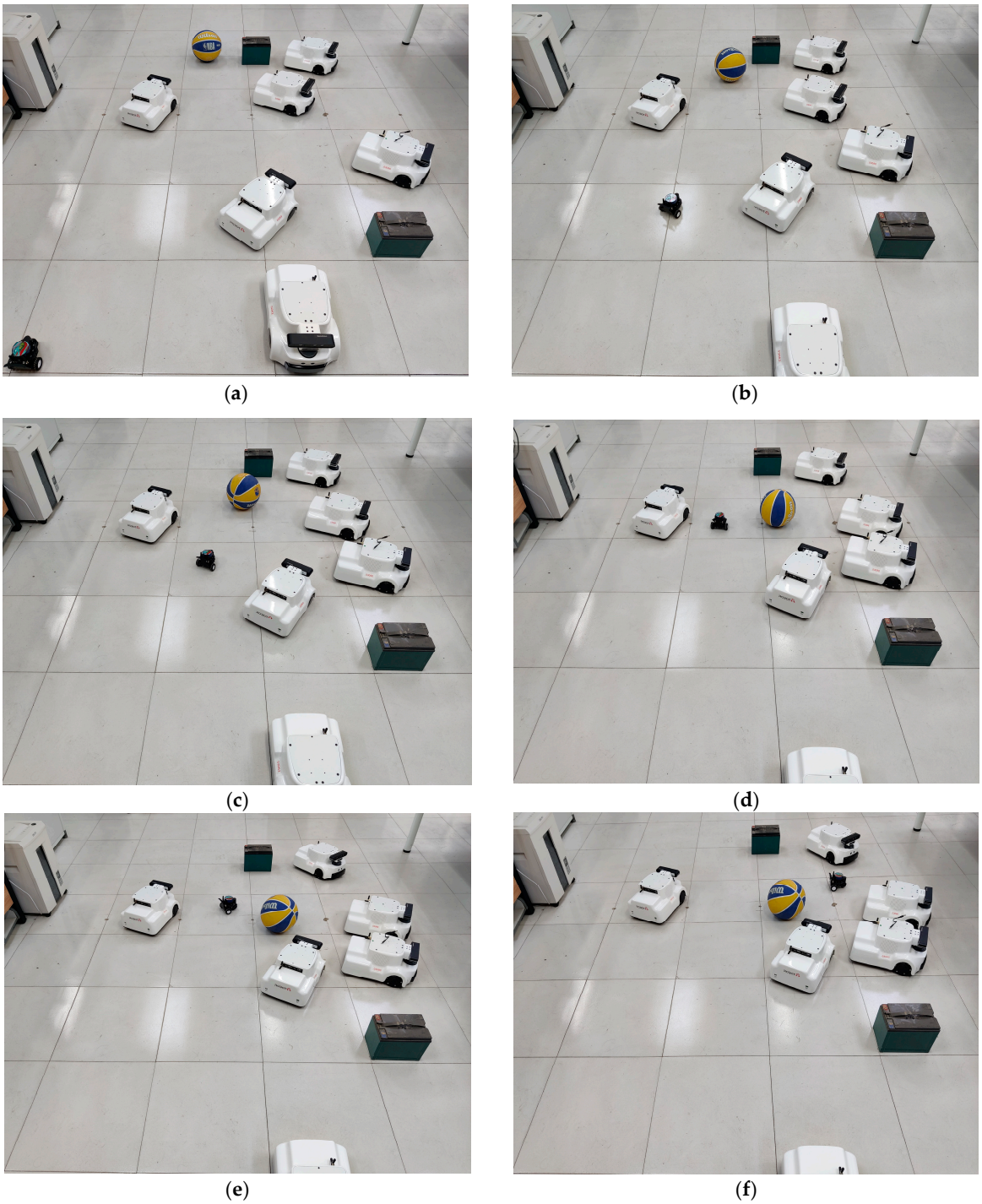


Figure 9. RRT*-FDWA path planning diagram. (**a**) Position planning diagram at the moment t = 1. (**b**) Position planning diagram at the moment t = 2. (**c**) Position planning diagram at the moment t = 5. (**d**) Position planning diagram at the moment t = 7.

Figure 10 shows the RRT*-FDWA experimental path planning diagram, and the experimental results show that the algorithm is effective in practical applications. The RRT*-FDWA algorithm was packaged into the path planner in the ROS navigation package before experimental validation. In the figure, the basketball and the white mobile robots are treated as moving obstacles and the experimental environment was the same as the simulation experiment. In the real experimental process, there was an error in the path planning process due to the robot control drive factor.



(**a**) (**b**) (**c**) (**d**) (**e**) (**f**)

Figure 10. *Cont.*

(**g**) (**h**)

Figure 10. RRT*-FDWA real environment path planning map. (**a**) The robot position map at the moment t = 0. (**b**) The robot position map at the moment t = 3. (**c**) The robot position map at the moment t = 5. (**d**) The robot position map at the moment t =7. (**e**) The robot position map at the moment t = 9. (**f**) The robot position map at the moment t = 13. (**g**) The robot position map at the moment t = 16. (**h**) The robot position map at the moment t = 18.

The RRT*-FDWA algorithm proposed in this paper was compared to the hybrid A*—an improved DWA algorithm—the hybrid algorithm being SOTA. Figure 11 shows the final completed path planning graph for both algorithms, the red dot [1,1] is the starting point and the green dot [9,9] is the target point; it can be seen that the A*-DWA algorithm still falls into local minima, while the RRT*-FDWA algorithm can complete the path planning more smoothly.

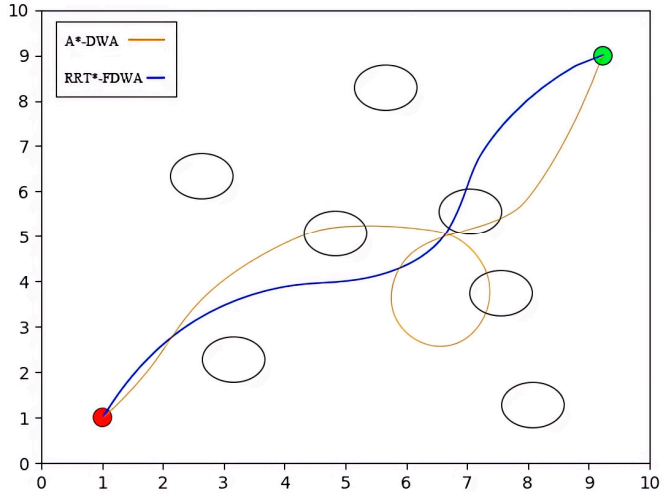


Figure 11. Algorithm comparison procedure.

It can be seen from Table 3 that the σ_w^2 of the RRT*-FDWA algorithm is smaller and δ_r is 12% higher than those of the A*-DWA algorithm, and it takes less time to complete. This indicates that the RRT*-FDWA algorithm is safer, fits better, and plans faster.

Table 3. Algorithm efficiency test comparison.

	σ_w^2	Time (s)	δ_r
A*-DWA	0.548	11.93	79%
RRT*-FDWA	0.424	9.62	91%

The general comparison of the four algorithms is given in Table 4. The RRT*-FDWA algorithm proposed in this paper considers both global and local optimality and can successfully plan a collision-free optimal path in both dynamic and static environments. In Table 4, E means Exist, N means None, L means Low, and H means High.

Table 4. Algorithm comparison.

	Dynamic Obstacle Avoidance	Local Optimality	Global Optimality	Smooth Path
RRT*	N	N	E	L
DWA	N	E	N	L
Fuzzy-DWA	E	E	N	H
RRT*-FDWA	E	E	E	H

6. Conclusions and Future Work

In this paper, mobile robot path planning in a dynamic environment was studied and an RRT*-FDWA algorithm was proposed. First, the RRT* algorithm was used for global path planning to obtain a global optimal route. The uncertainty of moving obstacles increases the difficulty of obstacle avoidance in the path planning process. When obstacles are encountered, the FDWA algorithm is added to improve robot adaptability in the face of a dynamic environment. The combination of the algorithms enables better avoidance of local minima and global replanning according to the new environment after the local planning has been completed. As a result, the robot has good maneuvering magnitude and safety and can complete the task in a shorter time.

The future goal is to apply this algorithm in multi-robot formation path planning to enable multi-robot formations with dynamic obstacle avoidance.

Author Contributions: Conceptualization, L.Z. and N.W.; methodology, N.W.; software, H.C.; validation, L.Z., N.W., and H.C; formal analysis, H.C.; investigation, N.W.; resources, Q.W.; data curation, L.Z.; writing—original draft preparation, N.W.; writing—review and editing, H.C.; visualization, Y.L.; project administration, L.Z.; funding acquisition, Q.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Henan Province Key Science and Technology Program (222102220085); the Henan Province University Key Science and Technology Project (23A413007); the Henan Province Key Science and Technology Project (222102210084); and the Henan Province Postgraduate Education Reform and Quality Improvement Project (YJS2023JD67), respectively.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Tan, G.; He, H.; Aaron, S. Global optimal path planning for mobile robot based on improved Dijkstra algorithm and ant system algorithm. *J. Cent. South Univ. Technol.* **2006**, *13*, 80–86. [CrossRef]
2. Chand, P.; Carnegie, D.A. A two-tiered global path planning strategy for limited memory mobile robots. *Robot. Auton. Syst.* **2012**, *60*, 309–321. [CrossRef]
3. Song, X.; Gao, S.; Chen, C.B.; Cao, K.; Huang, J. A new hybrid method in global dynamic path planning of mobile robot. *Int. J. Comput. Commun. Control* **2018**, *13*, 1032–1046. [CrossRef]
4. Persson, S.M.; Sharf, I. Sampling-based A* algorithm for robot path-planning. *Int. J. Robot. Res.* **2014**, *33*, 1683–1708. [CrossRef]
5. Kanayama, Y.J.; Hartman, B.I. Smooth local-path planning for autonomous vehicles1. *Int. J. Robot. Res.* **1997**, *16*, 263–284. [CrossRef]

6. Sedighi, K.H.; Ashenayi, K.; Manikas, T.W.; Wainwright, R.L.; Tai, H.-M. Autonomous local path planning for a mobile robot using a genetic algorithm. In Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No. 04TH8753), Portland, OR, USA, 19–23 June 2004; IEEE: Piscataway, NJ, USA, 2004; Volume 2, pp. 1338–1345.
7. Henkel, C.; Bubeck, A.; Xu, W. Energy efficient dynamic window approach for local path planning in mobile service robotics. *IFAC-PapersOnLine* **2016**, *49*, 32–37.
8. Karaman, S.; Frazzoli, E. Sampling-based algorithms for optimal motion planning. *Int. J. Robot. Res.* **2011**, *30*, 846–894. [CrossRef]
9. Karaman, S.; Frazzoli, E. Optimal kinodynamic motion planning using incremental sampling-based methods. In Proceedings of the 49th IEEE Conference on Decision and Control (CDC), Atlanta, GA, USA, 15–17 December 2010; pp. 7681–7687.
10. Karaman, S.; Walter, M.R.; Perez, A.; Frazzoli, E.; Teller, S. Anytime motion planning using the rrt*. In Proceedings of the 2011 IEEE International Conference on Robotics and Automation, Shanghai, China, 9–13 May 2011; pp. 1478–1483.
11. Kang, J.-G.; Jung, J.-W. Post Triangular Rewiring Method for Shorter RRT Robot Path Planning. *Int. J. Fuzzy Log. Intell. Syst.* **2021**, *21*, 213–221. [CrossRef]
12. Moon, C.; Chung, W. Kinodynamic planner dual-tree RRT (DT-RRT) for two-wheeled mobile robots using the rapidly exploring random tree. *IEEE Trans. Ind. Electron.* **2014**, *62*, 1080–1090. [CrossRef]
13. Nasir, J.; Islam, F.; Malik, U.; Ayaz, Y.; Hasan, O.; Khan, M.; Muhammad, M.S. RRT*-SMART: A rapid convergence implementation of RRT. *Int. J. Adv. Robot. Syst.* **2013**, *10*, 299. [CrossRef]
14. Chen, L.; Shan, Y.; Tian, W.; Li, B.; Cao, D. A fast and efficient double-tree RRT*-like sampling-based planner applying on mobile robotic systems. *IEEE/ASME Trans. Mechatron.* **2018**, *23*, 2568–2578. [CrossRef]
15. Qi, J.; Yang, H.; Sun, H. MOD-RRT*: A sampling-based algorithm for robot path planning in dynamic environment. *IEEE Trans. Ind. Electron.* **2020**, *68*, 7244–7251. [CrossRef]
16. Molinos, E.J.; Llamazares, A.; Ocaña, M. Dynamic window based approaches for avoiding obstacles in moving. *Robot. Auton. Syst.* **2019**, *118*, 112–130. [CrossRef]
17. Rasekhipour, Y.; Khajepour, A.; Chen, S.K.; Litkouhi, B. A potential field-based model predictive path-planning controller for autonomous road vehicles. *IEEE Trans. Intell. Transp. Syst.* **2016**, *18*, 1255–1267. [CrossRef]
18. Choi, Y.-I.; Cho, J.-H.; Kim, Y.-T. Collision Avoidance Algorithm of Mobile Robots at Grid Map Intersection Point. *Int. J. Fuzzy Log. Intell. Syst.* **2020**, *20*, 96–104. [CrossRef]
19. Kumar, S.; Parhi, D.R.; Muni, M.K.; Pandey, K.K. Optimal path search and control of mobile robot using hybridized sine-cosine algorithm and ant colony optimization technique. *Ind. Robot.* **2020**, *47*, 535–545. [CrossRef]
20. Guo, B.; Guo, N.; Cen, Z. Obstacle avoidance with dynamic avoidance risk region for mobile robots in dynamic environments. *IEEE Robot. Autom. Lett.* **2022**, *7*, 5850–5857. [CrossRef]
21. Zhong, X.Y.; Peng, X.F.; Zhou, J.H.; Huang, X.C. Mobile robot path planning based on local environment modeling and adaptive window. *Appl. Mech. Mater.* **2011**, *48*, 679–683. [CrossRef]
22. Chang, L.; Shan, L.; Jiang, C.; Dai, Y. Reinforcement based mobile robot path planning with improved dynamic window approach in unknown environment. *Auton. Robot.* **2021**, *45*, 51–76. [CrossRef]
23. Xiang, L.; Li, X.; Liu, H.; Li, P. Parameter Fuzzy Self-Adaptive Dynamic Window Approach for Local Path Planning of Wheeled Robot. *IEEE Open J. Intell. Transp. Syst.* **2021**, *3*, 1–6. [CrossRef]
24. Wu, B.; Chi, X.; Zhao, C.; Zhang, W.; Lu, Y.; Jiang, D. Dynamic Path Planning for Forklift AGV Based on Smoothing A* and Improved DWA Hybrid Algorithm. *Sensors* **2022**, *22*, 7079. [CrossRef] [PubMed]
25. Trinh, L.A.; Ekström, M.; Cürüklü, B. Petri net based navigation planning with dipole field and dynamic window approach for collision avoidance. In Proceedings of the 2019 6th International Conference on Control, Decision and Information Technologies (CoDIT), Paris, France, 23–26 April 2019; pp. 1013–1018.
26. Zeng, T.; Si, B. Mobile robot exploration based on rapidly-exploring random trees and dynamic window approach. In Proceedings of the 5th International Conference on Control, Automation and Robotics (ICCAR), Beijing, China, 19–22 April 2019; pp. 51–57.
27. Fox, D.; Burgard, W.; Thrun, S. The Dynamic Window Approach to Collision Avoidance. *IEEE Robot. Autom. Mag.* **1997**, *4*, 23–33. [CrossRef]