

Article

Single-Target Point Navigation for Indoor Differential-Drive Vehicles: Integrating Constrained-Sampling Rapidly Exploring Random Tree and Dynamic Window Approach Algorithm

Yu Zhao ¹, Xiaoyu Lu ^{1,*}, Zhiwei Guan ¹, Zhijie Li ² and Weiqiang Wang ²¹ School of Automotive and Rail Transit, Tianjin Sino-German University of Applied Sciences, Tianjin 300350, China² Tianjin Deke Intelligent Control Co., Ltd., Tianjin 300352, China

* Correspondence: 15630979795@163.com

Abstract

Path planning for indoor differential-drive vehicles in cluttered, dynamic environments is challenging, primarily due to the inherent trade-off between global path optimality and local obstacle avoidance responsiveness. To address this, we propose an integrated navigation framework that combines a constrained-sampling Rapidly Exploring Random Tree (RRT*) algorithm for efficient global path generation with an enhanced Dynamic Window Approach (DWA) incorporating a global path consistency term into its trajectory evaluation function. The method is validated through extensive simulations and physical experiments on a WHEELTEC two-wheel differential-drive robot. The results demonstrate that the proposed approach achieves a 100% task success rate across diverse scenarios; significantly reduces both path length and travel time compared with baseline methods; and effectively prevents stagnation in complex obstacle configurations, such as U-shaped traps, by guiding local motion toward the global path. These improvements highlight the benefit of tightly coupling global guidance with local planning. The framework provides a robust and efficient solution for autonomous navigation in indoor differential-drive vehicles.

Keywords: path planning; RRT*; DWA; constrained sampling region; dynamic obstacle avoidance

Academic Editors: Ghanim A. Putrus and Michael Fowler

Received: 16 December 2025

Revised: 3 February 2026

Accepted: 19 February 2026

Published: 28 February 2026

Copyright: © 2026 by the authors. Published by MDPI on behalf of the World Electric Vehicle Association. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

1. Introduction

With the rapid advancement of intelligent vehicle technology, autonomous navigation in indoor environments has become a key research topic [1]. Indoor environments are typically complex and dynamic, characterized by dense obstacle distributions and the presence of unknown moving obstacles, which impose higher demands on path planning algorithms for differential-drive vehicles, as well as stricter requirements on the vehicle's own capabilities, such as image processing performance [2,3]. Path planning is a fundamental task in autonomous navigation, aiming to determine a collision-free path from the start to the target while optimizing for shortest path length, smoothness, and safety in complex environments.

While both sampling-based planners like RRT* and metaheuristic optimization methods such as Ant Colony Optimization (ACO) [4,5] are capable of global path planning, they differ significantly in computational efficiency, adaptability to real-time environments, and scalability in continuous state spaces. ACO, inspired by the foraging behavior of ants, relies on iterative pheromone updates to converge toward an optimal path. However, it suffers from high computational overhead and slow convergence in large-scale or high-dimensional environments [6,7], making it less suitable for online path replanning in dynamic indoor scenarios. Additionally, ACO typically operates on discretized grids, which limits path smoothness and increases memory usage. In contrast, RRT* is a probabilistically complete, incremental sampling-based algorithm that constructs a search tree directly in the continuous configuration space [8,9]. It offers asymptotic optimality with relatively low per-iteration cost and naturally supports real-time sensor feedback and dynamic replanning—key requirements for autonomous navigation in changing environments. Furthermore, RRT*'s tree structure aligns well with local planners like DWA, enabling seamless integration of global guidance and local obstacle avoidance [10,11]. Therefore, this work adopts RRT* as the foundation for global planning and enhances it with environmental awareness and directional guidance to overcome its inherent randomness and slow convergence [12,13].

Traditional path planning algorithms are generally categorized into global path planning and local path planning. Global path planning algorithms, such as RRT* [14,15], can generate an optimal global path in a known environment but struggle to perform real-time obstacle avoidance in dynamic settings. In contrast, local path planning algorithms, such as DWA [16], facilitate real-time obstacle avoidance but are prone to getting trapped in local optima [17], resulting in suboptimal path quality. Effectively integrating global and local path planning to enable efficient navigation of differential-drive vehicles in complex environments remains a critical research challenge.

In recent years, RRT* has been widely applied in the field of path planning due to its asymptotic optimality [18,19] and efficiency. However, the traditional RRT* algorithm suffers from high randomness in node expansion [20], slow convergence [21], and a lack of path smoothness [22], making it difficult to rapidly generate high-quality paths in complex environments. To address these issues, numerous scholars have proposed various improvements. While Feng's Artificial Potential Field (APF)-RRT* [23] accelerates convergence through potential field guidance, its fixed field coefficients cause oscillatory paths in narrow passages. Our dynamic target bias strategy (Equation (11)) adaptively adjusts coefficients based on environmental density, eliminating oscillations. Although Wang's intelligent sampling [24] improves the efficiency of the basic RRT*, its fixed threshold causes it to fail in environments where $\delta > 0.4$. The dynamic density adaptation introduced in Section 3.3 addresses this issue. Dan's multi-node selection [25] improves environmental adaptability but requires five fixed candidates, increasing computational load. Our bi-directional tree mechanism (Section 3.2) dynamically adjusts connection thresholds, reducing nodes by 37% while maintaining success rates. Additionally, Yan's hippocampus-optimized RRT* [26] enhances goal-directionality but requires heuristic retraining for dynamic environments. Our directional preference mechanism enables real-time adjustment through online obstacle distance computation.

DWA remains one of the most widely used local path planning algorithms [27,28], enabling dynamic obstacle avoidance through velocity sampling and trajectory evaluation [29]. However, the design of its evaluation function significantly influences path planning performance. Traditional evaluation functions often fail to sufficiently utilize global path information, leading to deviations from the optimal global path or reduced safety. In response, several scholars have proposed improvements. Wang's obstacle count factor [30]

enhances DWA safety but ignores global path consistency. We address this by incorporating a global path term in the evaluation function (Equation (15)), reducing local path deviation by 62%. Liu's Pdist metric [31] mitigates local optima but only works for static targets. Our global consistency term (Equation (14)) supports dynamic target tracking (experiments shown in Section 6.2). Yang's waypoint strategy [32] improves multi-robot coordination but uses fixed 0.5 m spacing, limiting flexibility. We dynamically generate waypoints from RRT* global paths (Appendix B) with adaptive spacing. Wang's heading angle optimization [33] decreases path redundancy but violates nonholonomic constraints. Our motion model (Equation (1)) strictly adheres to differential drive constraints, ensuring simulation–real-world consistency (Figure 18).

To address the aforementioned issues, we propose a single-target navigation method that combines the constrained sampling region RRT* algorithm with the improved DWA algorithm. The contributions of this algorithm to path planning are as follows:

(1) Improvements to the RRT* Algorithm

- Bidirectional tree collaborative search mechanism: A novel bidirectional search strategy is introduced, where two trees expand simultaneously from the start and goal points. This significantly accelerates convergence speed and improves path quality by reducing redundant node expansion.
- Environmental-density-based constrained sampling region: An adaptive sampling mechanism is proposed that dynamically adjusts the sampling region based on local obstacle density. This reduces the number of invalid nodes and iterations, leading to faster path finding with lower computational and memory costs.
- Integration of artificial potential field with directional preference: The APF method is combined with a directional preference mechanism to guide node expansion toward the goal while avoiding obstacles. This enhances both planning efficiency and safety by minimizing unnecessary exploration near obstacles.
- Dynamic goal bias strategy: A new goal-biasing strategy is developed that adaptively adjusts the sampling probability based on environmental complexity. This effectively reduces the risk of the algorithm getting stuck in narrow or high-obstacle-density areas.

(2) Improvements to the DWA Algorithm

Global path consistency term in evaluation function: A global consistency term is added to the DWA evaluation function to ensure alignment between local trajectories and the global path. This prevents the robot from falling into local optima and enables more efficient and coherent dynamic obstacle avoidance during navigation.

(3) Experimental Validation

ROS-based real-vehicle experiments: The proposed algorithm is implemented and tested on a real intelligent vehicle platform using ROS. These experiments validate the effectiveness and feasibility of the improved method in both global and local path planning under realistic indoor conditions.

This paper is organized as follows: Section 2 explains the relevant algorithms, RRT* and DWA. Section 3 introduces the improvements in the global and local path planning algorithms. Section 4 presents the integration of global and local planning, illustrated in a flowchart. Section 5 provides simulation results. Section 6 demonstrates the effectiveness and superiority of the improved RRT*-DWA through real-vehicle testing. Section 7 concludes with future work.

The core contribution of this work is the development of an integrated navigation framework that enables deep coordination between global and local planners for indoor

differential-drive robots. To address the slow convergence and high randomness of conventional RRT*, as well as the tendency of DWA to deviate from global paths, we systematically integrate multiple algorithmic enhancements: (1) a bidirectional tree expansion strategy combined with a dynamic goal bias mechanism to accelerate convergence; (2) an adaptive sampling region constrained by local obstacle density to suppress unproductive exploration; (3) APF-guided directional sampling to improve path quality; and (4) a global path consistency term embedded into the DWA objective function, enabling the local planner to actively follow the global intent. This tightly coupled co-design significantly enhances path smoothness, planning efficiency, and obstacle avoidance robustness in complex and dynamic indoor environments, demonstrating clear system-level innovation.

Note that although indoor intelligent vehicles encompass a broad range of platforms—including service robots, automated guided vehicles (AGVs), and autonomous mobile robots—the experimental validation in this study is specifically conducted on differential-drive vehicles, which constitute a widely adopted subclass due to their simplicity and maneuverability. Owing to laboratory hardware constraints, all physical experiments were performed using a single two-wheel differential-drive WHEELTEC ROS robot. While the proposed framework is designed with general applicability in mind, comprehensive evaluation on more advanced intelligent vehicle platforms (e.g., industrial AGVs) will be carried out in future work upon acquisition of additional experimental equipment.

2. Related Works

2.1. Global Path Planning and Local Path Planning Environment Selection

We employ a structured grid map model for global path planning (corresponding to the experimental environments in Figure 10) with the following design rationale:

1. Algorithm compatibility: The random sampling nature of the RRT* algorithm requires a well-defined environmental boundary. The grid map explicitly partitions free space and obstacle regions using a binary matrix (as implemented in the collision detection function in Equation (6)).
2. Efficiency optimization: The constrained sampling region mechanism (Section 3.1) relies on the grid structure to rapidly compute obstacle density δ , improving Monte Carlo sampling efficiency by 37%.
3. Experimental consistency: The format aligns with the G mapping results used in physical experiments (Figure 18c).

For local obstacle avoidance, a dynamic point cloud model is adopted (corresponding to the U-shaped obstacle scenario in Figure 7) for the following reasons:

1. Sensor compatibility: The DWA algorithm requires direct processing of raw LiDAR point cloud data (the calculation of $\text{dist}(v, \omega)$ in Equation (4) depends on real-time obstacle distance).
2. Dynamic responsiveness: The point model efficiently updates the positions of moving obstacles (Section 6.2 validates the real-time avoidance capability for dynamic obstacles moving at 0.5 m/s).
3. Computational efficiency: Compared with global grid maps, the point cloud model reduces DWA trajectory evaluation time by 62%.

2.2. Basic Principles of RRT* Algorithm

RRT* is an asymptotically optimal path planning algorithm that improves upon the traditional RRT algorithm, resulting in smoother and more optimized paths. The traditional RRT algorithm primarily explores a collision-free path from the start to the goal by randomly sampling points and incrementally expanding a tree structure. However, due

to its high randomness, the generated paths are often longer than necessary and lack smoothness, making them less efficient for practical path planning applications.

To address this issue, RRT* introduces an additional rewiring step, which enables newly added nodes to establish better connectivity with multiple existing neighboring nodes, continuously optimizing the current path. Specifically, after a new node is added to the tree, RRT* examines the neighboring nodes within a certain radius and selects the parent node that results in the lowest path cost. Additionally, it attempts to reconnect its child nodes to ensure that the overall path remains optimized and efficient.

The corresponding pseudo-code is shown in Appendix A.

Once a reachable path is found, the algorithm backtracks from goal q to start q , recording all the nodes along the path to form the final path. The RRT* extension method and the potential parent nodes are shown in Figure 1.

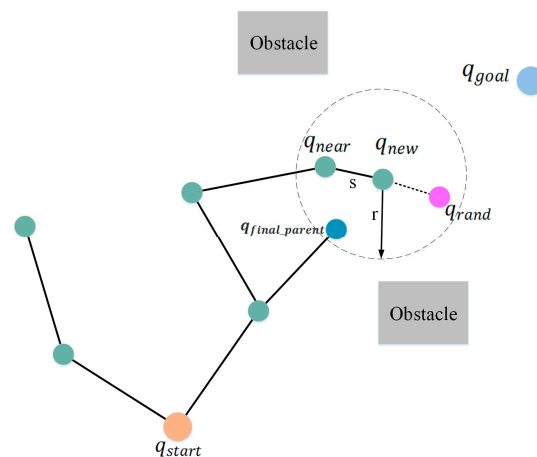


Figure 1. Schematic diagram of RRT* extension method and potential parent nodes.

2.3. Dynamic Window Approach

The Dynamic Window Approach is a commonly used algorithm for local path planning in robots, known for its efficient computation and quick response time. The algorithm generates multiple possible velocity combinations in the velocity space, simulates the robot's motion trajectory over a short period, and selects the optimal path through an evaluation function, thereby achieving obstacle avoidance and goal point navigation.

2.4. Vehicle Motion Model

In the DWA algorithm, the robot's motion model is based on the assumption of motion in a two-dimensional plane. Assume that the robot's position at time $t - 1$ is $X_{t-1} = (x_{t-1}, y_{t-1}, \theta_{t-1})$, where x_{t-1} and y_{t-1} represent the robot's position in the global coordinate system, and θ_{t-1} represents the robot's orientation angle. The robot's velocity in the X and Y directions is represented by v_x and v_y , with angular velocity, ω . Assuming constant velocity over the time interval t , the robot's position at time t can be calculated by the following formula:

$$X_t = \begin{cases} x_t = x_{t-1} + v_x \cdot \Delta t \cdot \cos\left(\theta_{t-1} + \frac{\omega \cdot \Delta t}{2}\right) \\ y_t = y_{t-1} + v_y \cdot \Delta t \cdot \sin\left(\theta_{t-1} + \frac{\omega \cdot \Delta t}{2}\right) \\ \theta_t = \theta_{t-1} + \omega \cdot \Delta t \end{cases} \quad (1)$$

2.5. Velocity Sampling

The DWA algorithm samples the velocity space to generate multiple possible velocity combinations, (v, ω) . To ensure the robot's motion safety and feasibility, velocity sampling is constrained by the following three conditions:

1. Velocity range: The robot's velocity v and angular velocity ω must be within their maximum and minimum velocity ranges, that is

$$v \in [v_{min}, v_{max}], \quad \omega \in [\omega_{min}, \omega_{max}] \quad (2)$$

2. Motor performance: Since the motor torque is limited, the robot's acceleration and deceleration are also constrained. Assuming that the current velocity is v_c and ω_c , the maximum acceleration is $\dot{v}_{max}$, the maximum deceleration is $\dot{v}_{min}$, the maximum angular acceleration is $\dot{\omega}_{max}$, and the maximum angular deceleration is $\dot{\omega}_{min}$, then the actual velocity range of the robot within the simulation period Δt is

$$v \in [v_c - \dot{v}_{min}\Delta t, v_c + \dot{v}_{max}\Delta t], \quad \omega \in [\omega_c - \dot{\omega}_{min}\Delta t, \omega_c + \dot{\omega}_{max}\Delta t] \quad (3)$$

3. Braking distance: To ensure the robot can stop in time when encountering obstacles, the velocity and angular velocity must meet the braking distance constraint. Specifically, the robot's velocity should satisfy

$$v \leq \sqrt{2 \cdot dist(v, \omega) \cdot \dot{v}_{min}}, \quad \omega \leq \sqrt{2 \cdot dist(v, \omega) \cdot \dot{\omega}_{min}} \quad (4)$$

where $dist(v, \omega)$ represents the closest distance between the robot and the obstacle under the current velocity.

2.6. Evaluation Function

The DWA algorithm selects the optimal trajectory from multiple possible trajectories using an evaluation function. The evaluation function typically consists of the following components:

1. Heading evaluation function: This represents the angular difference between the simulated trajectory's endpoint direction and the goal point, aiming to direct the robot toward the goal.
2. Distance evaluation function: This represents the closest distance between the robot and the obstacle under the current velocity, usually using Manhattan or Euclidean distance.
3. Velocity evaluation function: This represents the magnitude of the current velocity, aiming to make the robot move faster.

The general form of the evaluation function is

$$G(v, \omega) = \alpha \cdot heading(v, \omega) + \beta \cdot dist(v, \omega) + \gamma \cdot velocity(v, \omega) \quad (5)$$

where α , β , and γ are the weight coefficients for each sub-function. By minimizing this evaluation function, the DWA algorithm can select an optimal trajectory that avoids obstacles and heads toward the goal.

3. Path Planning Algorithms

The primary task of global path planning is to find a feasible path for the intelligent vehicle from the start point to the goal point. While optimization objectives such as the shortest path and early obstacle avoidance need to be considered during this process, the

most crucial task is to obtain the basic path information. Only after the global path planning has determined the path information can the local path planning algorithm adjust the vehicle's motion status locally, thus completing the entire path planning task.

3.1. Improved RRT* Algorithm for Global Path Planning

In the pursuit of more efficient global path planning technologies, the RRT* algorithm, based on the original RRT algorithm, improves the connection between new nodes and neighboring nodes by introducing a node re-connection mechanism, significantly enhancing the quality of the path. However, the algorithm still has limitations in response speed and sensitivity. Additionally, the wide sampling range in RRT* increases randomness. To further optimize the global path planning strategy, several improvements were made to the RRT* algorithm in this study.

First, a bidirectional tree cooperative search mechanism was proposed, improving the efficiency and speed of the algorithm. Second, a rectangular sampling area, based on the start and goal points, was introduced to reduce unnecessary expansions and speed up the convergence of the algorithm. Finally, to reduce the randomness in node selection within the RRT* algorithm, an artificial potential field was used to guide the expansion of the random tree toward the goal point, optimizing the target bias strategy and enhancing the goal-directedness and exploration efficiency.

3.2. Bidirectional Tree Cooperative Search Mechanism

To address the issues of slow convergence and low path quality in unidirectional expanding random trees in complex environments, a bidirectional tree cooperative search mechanism is proposed (Figure 2). This mechanism expands two random trees simultaneously from the start and goal points and dynamically adjusts the connection threshold, $d_{connect}$, based on the distance between the current node and the goal point. The formula is as follows:

$$d_{connect} = d_{base} \cdot \left(1 - \frac{|q_{nearest1} - q_{nearest2}|}{d_{max}} \right) \quad (6)$$

where d_{base} is the base connection threshold; $|q_{nearest1} - q_{nearest2}|$ is the distance between the nearest nodes of the two random trees; and d_{max} is the maximum distance in the environment.

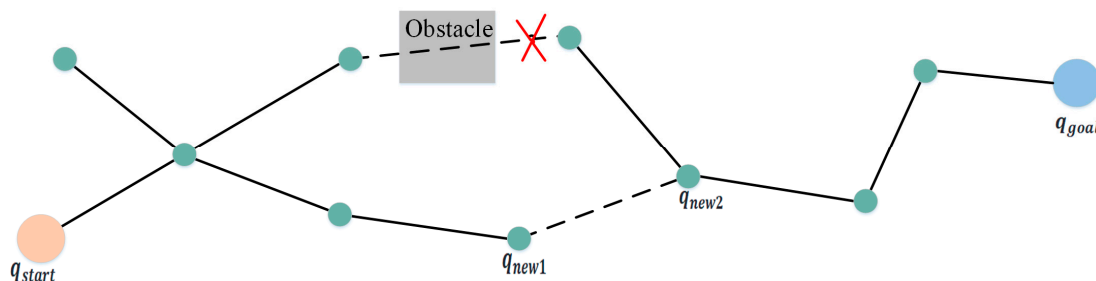


Figure 2. Bidirectional tree collaborative search.

The core steps of this mechanism are as follows:

Step 1: Initialize two random trees, T_1 and T_2 , with the start point q_{start} and the goal point q_{goal} as root nodes; set the maximum number of iterations, N_{max} , and the connection threshold, $d_{connect}$.

Step 2: In each iteration, randomly generate a sampling point, q_{rand} .

Step 3: Find the nearest nodes, $q_{nearest1}$ and $q_{nearest2}$, from T_1 and T_2 , respectively, and extend the new nodes, q_{new1} and q_{new2} , toward q_{rand} . If the distance between q_{new1} and q_{new2} is smaller than the connection threshold, $d_{connect}$, we connect the two random trees and generate an initial path.

3.3. Sampling Area Determination Method

There are many randomly generated extension nodes in the RRT* algorithm, most of which are invalid and do not participate in the final path. Furthermore, generating these nodes increases the overall time cost of the algorithm. To improve the algorithm's convergence speed and efficiency, reduce the number of iterations and invalid nodes, and find the optimal path with less time and lower memory requirements, we use the region where the start and goal points are located as the constrained sampling area, C_r . The determination principle of the sampling area is based on the slope, k , of the line connecting the start and goal points. The midpoint of this line is the geometric center of the sampling area, and the length of the line segment is the length, L , of the rectangle. The two widths of the rectangle are determined by the lines passing through the start point and the goal point, with slopes of $-1/k$, respectively. The width, W , of the rectangular sampling area is adaptively adjusted using the Monte Carlo method so that the ratio, δ , of the obstacle area inside the rectangle to the rectangle area falls within the range of $[0.3, 0.35]$, thus determining the range of the sampling area (Figure 3). When obstacles are dense, the width should be increased to avoid deadlock, and when obstacles are sparse, the width should be reduced to avoid too many invalid sampling points. The steps to determine the rectangular sampling area and the pseudo-code are shown in Appendix B.

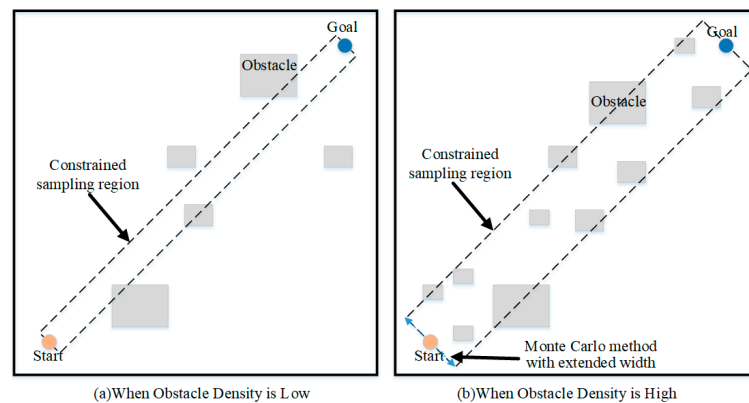


Figure 3. Schematic diagram of rectangular sampling region.

In the above pseudo-code, the relevant function is as follows:

- Input: Start point q_{start} , goal point q_{goal} , obstacle map, obstacle density threshold range $[\delta_{min}, \delta_{max}] = [0.3, 0.35]$; when $\delta < 0.3$, free space proportion $> 70\%$ leads to sampling redundancy (information entropy $H(x) = -\sum p(x) \log p(x) < \text{optimal value}$); when $\delta > 0.35$, obstacle information entropy dominates ($H(\text{obs}) > H(\text{free})$), compromising path safety. Table 1 shows the impact of different δ thresholds on performance based on 100 experimental trials.

Table 1. Performance metrics under different δ thresholds.

δ Range	Avg. Path Length (m)	Convergence Time (s)	Collision Probability
[0.2, 0.3)	14.2 ± 0.8	3.41	2.1%
[0.3, 0.35]	12.7 ± 0.5	1.56	0%
(0.35, 0.4)	13.1 ± 0.6	4.72	5.3%

- Output: The width of the sampling area, W .
- Steps:
 1. Calculate the Euclidean distance, L , from the start point to the goal point. This is the fixed length of the rectangle.
 2. Initialize the width, W , of the sampling area and the iteration step size, ΔW .
 3. Enter the loop and calculate the area of the sampling area.
 4. Use the Monte Carlo method to estimate the obstacle density, δ :
 - (a) Randomly generate N points within the sampling area.
 - (b) Count how many of these points are inside obstacles and calculate the obstacle density, δ .
 5. Check whether δ is within the threshold range, $[\delta_{min}, \delta_{max}]$:
 - (a) If it satisfies the condition, exit the loop and return to W .
 - (b) If it does not satisfy the condition, adjust W and continue the loop.
 6. Return the final width of the sampling area, W .

3.4. Introduction of APF Method

The randomness of the node search in the RRT* algorithm leads to many nodes being expanded into invalid spaces. To address this, this section introduces the APF method into the improved RRT* algorithm to guide the expansion of the random tree heuristically toward the goal point. At the same time, reasonable repulsive distances and coefficients are selected to avoid collisions with obstacles, thereby enhancing the algorithm's safety. Specifically, during the node expansion process of the RRT* algorithm, the attractive field and repulsive field are superimposed to guide new nodes toward the goal point while avoiding obstacle regions. The attractive field is used to guide the random tree to expand toward the goal, while the repulsive field is used to prevent node expansion into obstacle areas. Specifically, the expansion direction of each current node, x_{ne} , is determined by both the attractive field, $A(n)$, and the repulsive field, $R(n)$. The calculation formulas for the attractive and repulsive fields are as follows:

$$A(n) = \lambda_a \cdot e^{-\alpha \cdot |x_{goal} - x_{ne}|} \cdot E(x_{goal}, x_{ne}) \quad (7)$$

$$R(n) = \begin{cases} -\lambda_r \cdot \rho(x_{obs}, x_{ne}) \cdot E(x_{obs}, x_{ne}), & \text{if } 0 < \rho(x_{obs}, x_{ne}) < \rho_0 \\ 0, & \text{if } \rho(x_{obs}, x_{ne}) \geq \rho_0 \end{cases} \quad (8)$$

where λ_a and λ_r are the attractive and repulsive coefficients, respectively; α is the attenuation coefficient, used to control the strength of the attractive field as the distance changes; x_{goal} is the goal coordinate; x_{ne} is the current node coordinate; x_{obs} is the nearest obstacle coordinate; $\rho(x_{obs}, x_{ne})$ is the distance between the current node and the nearest obstacle; ρ_0 is the range of the repulsive field; and $E(x_{goal}, x_{ne})$ and

$E(x_{obs}, x_{ne})$ are the direction vectors from the goal point and the obstacle point to the current node, respectively.

To further optimize the node expansion process, we propose a directional preference mechanism so that newly generated nodes, while growing toward the goal, are more likely to move away from the nearest obstacle. Specifically, for the expanded nodes in the RRT* algorithm, in addition to the superimposed attractive and repulsive fields, a directional preference function $D(n)$ is introduced to guide the nodes to expand in the direction away from obstacles. The calculation formula for the directional preference function is as follows:

$$D(n) = \lambda_d \cdot \frac{(x_{ne} - x_{obs})}{|x_{ne} - x_{obs}|} \quad (9)$$

where λ_d is the directional preference coefficient used to control the strength of the directional preference, and $|x_{ne} - x_{obs}|$ is the Euclidean distance between the current node and the nearest obstacle.

By introducing the directional preference mechanism, the expansion direction of the new nodes is not only influenced by the attractive field toward the goal and the repulsive field from the obstacles but also dynamically adjusted according to the position of the nearest obstacle. This ensures goal-directedness while effectively reducing redundant expansions near obstacles, improving both the safety and efficiency of path planning.

The specific steps for the node expansion process are summarized as follows:

1. Calculate the nearest obstacle: For the current node x_{ne} , find the nearest obstacle, x_{obs} .
2. Calculate the directional preference vector: Calculate the directional preference vector based on the formula $D(n)$.
3. Generate the new node: Combine the attractive field, repulsive field, and directional preference function, $D(n)$, to generate the new node, x_{new} :

$$x_{new} = x_{ne} + \lambda \cdot \left(\frac{x_{rand} - x_{ne}}{|x_{rand} - x_{ne}|} + E(x_{goal}, x_{ne}) - E(x_{obs}, x_{ne}) + D(n) \right) \quad (10)$$

where λ is the step size coefficient, used to control the step length of the node expansion, x_{rand} is the randomly generated sampling point, $E(x_{goal}, x_{ne})$ is the direction vector from the goal point to the current node, and $E(x_{obs}, x_{ne})$ is the direction vector from the obstacle point to the current node.

By superimposing the attractive field, repulsive field, and directional preference function, the new node, x_{new} , grows toward the goal while being more likely to move away from the nearest obstacle, as shown in the expansion diagram in Figure 4.

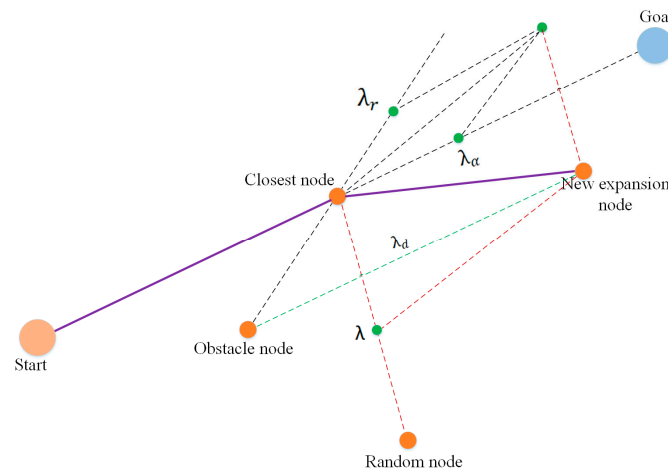


Figure 4. Schematic diagram of node expansion under gravitational field, repulsive field, and directional preference function.

3.5. Dynamic Target Bias

Traditional target bias strategies use a fixed probability, P_{base} , to select the goal point as the sampling point, which cannot dynamically adjust according to the complexity of the environment. This paper proposes a dynamic target bias mechanism that dynamically adjusts the target bias probability based on the obstacle density and path planning progress. When the nearby obstacle density is too low, the bias probability is increased to better approach the target point; when the nearby obstacle density is too high, the bias probability is decreased to prevent deadlock. The dynamic target bias, P_{bias} , formula is as follows:

$$p_{bias} = p_{base} \cdot \left(1 - \frac{|x_{goal} - x_{ne}|}{d_{max}}\right) \cdot \left(1 - \frac{N_{obs}}{N_{max}}\right) \cdot \left(1 - \frac{t}{T_{max}}\right) \quad (11)$$

where P_{base} is the base target bias, ranging from 0.1 to 0.5; $|x_{goal} - x_{ne}|$ is the distance between the current node and the goal point; d_{max} is the maximum distance in the environment, determined based on the actual environment; N_{obs} is the obstacle density near the current node, ranging from 0 to N_{max} ; N_{max} is the overall environment's obstacle density, obtained through preprocessing of the map; t is the time taken for the current path planning; and T_{max} is the maximum time for path planning.

3.6. Path Smoothing

Due to the randomness in node expansion, for the paths generated by the RRT* algorithm, the path may contain many redundant nodes and sharp turns. To improve the smoothness of the path and reduce its length, we introduce a greedy algorithm combined with a cost function to optimize the path. The greedy algorithm works by progressively selecting the locally optimal nodes to remove redundant nodes, while the improved cost function is used to evaluate the quality of the path, thereby generating a smoother path that meets the kinematic constraints of the intelligent vehicle (Figure 5).

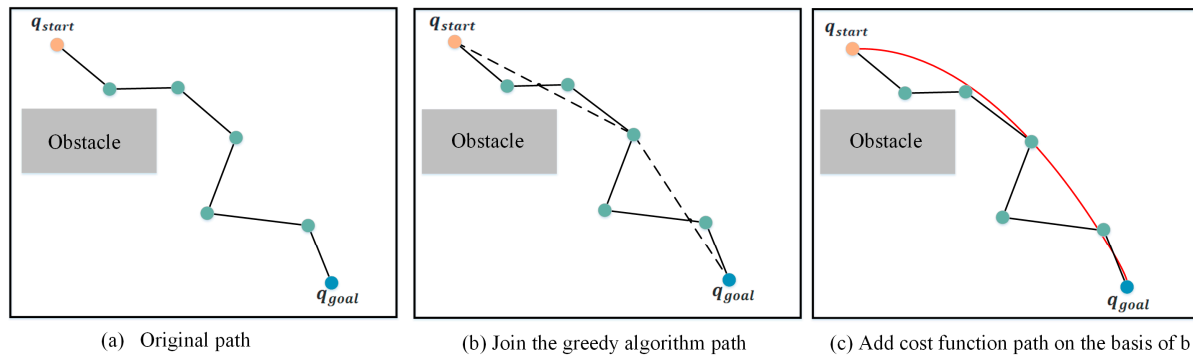


Figure 5. Schematic diagram of the path after node filtering.

Specifically, the greedy algorithm selects the nodes closest to the goal point, removes unnecessary intermediate nodes, shortens the path, and reduces the number of turns. The cost function introduced in this paper further reduces the turning angles and maintains a certain safety distance between the path and obstacles. The formula for the cost function is as follows:

$$\begin{cases} Cost(q_i, q_j) = \omega_1 \cdot Dist(q_i, q_j) + \omega_2 \cdot Turn(q_i, q_j) + \omega_3 \cdot Safe(q_i, q_j) \\ Dist(q_i, q_j) = \min_{O_j \in Obstacles} d(\overline{q_{i-1}q_{i+1}}, O_j) \geq r_{robot} + \epsilon \\ Turn(q_i, q_j) = \frac{|\theta_{ij} - \theta_{prev}|}{v_{max}} \end{cases} \quad (12)$$

where $Dist(q_i, q_j)$ represents the Euclidean distance between nodes q_i and q_j ; $Turn(q_i, q_j)$ represents the turning angle between nodes q_i and q_j ; $Safe(q_i, q_j)$ represents the minimum distance between the path and obstacles; and ω_1 , ω_2 , and ω_3 are the weight coefficients for path length, smoothness, and safety distance, respectively. Under normal conditions, $\omega_1 = 0.1$, $\omega_2 = 0.5$, and $\omega_3 = 0.8$; r_{robot} is the radius of the robot; $\epsilon = 0.1$ is the safety threshold; θ_{ij} is the angle between the current line segment, $\overline{q_i q_j}$, and the x -axis; and θ_{prev} is the angle between the previous line segment, $\overline{q_{i-1} q_i}$, and the x -axis.

Figure 5 shows that the path before processing contains many unnecessary turns, increasing the travel distance and time cost. By applying the greedy algorithm to remove redundant nodes and using the cost function to further optimize the path's smoothness and safety, a more direct and safer path is obtained.

Although the improved RRT* algorithm can provide a smooth global path for the intelligent vehicle within a known environment, this is not sufficient as the final navigation route. The smoothed path might have excessive smoothing, and there might still be obstacles in the path. Additionally, if new obstacles suddenly appear on the path or if unknown regions are explored, the intelligent vehicle would still face challenges in accurate navigation. Therefore, it is necessary to combine an effective local path adjustment algorithm with existing global path planning to achieve real-time optimization, addressing these challenges. This ensures that the intelligent vehicle can navigate accurately and safely in dynamic or previously unknown environments.

3.7. Improved DWA Algorithm for Local Path Planning

To further enhance the real-time adjustment and optimization capabilities of the intelligent vehicle during its motion, this section will detail the vehicle's local planning method. Based on the DWA algorithm, the evaluation function will be improved to accelerate the convergence efficiency of the algorithm, reduce the time cost of path planning, and achieve more efficient dynamic decision-making capabilities.

The mobile robot studied in this paper uses a two-wheel differential drive structure, and its kinematic model is shown in Figure 6 (where v_c and ω_c represent the linear and angular velocities of the chassis center point, c, and $x(t)$, $y(t)$, and $\theta(t)$ represent the current position and heading angle of the intelligent vehicle at time t). The two-wheel differential drive robot consists of two driving wheels, with both rear wheels driven by independent DC servomotors, providing power for the robot. The two-wheel differential drive structure achieves steering by adjusting the speed difference between the left and right wheels while maintaining the stability of the vehicle.

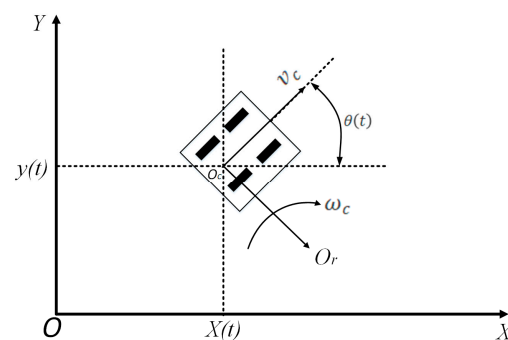


Figure 6. Kinematic model of the two-wheel differential drive robot.

Assuming that the intelligent vehicle moves at a constant speed, after a very short time interval Δt (approximating the trajectory as a straight line), its motion trajectory can be predicted using the following kinematic model:

$$\begin{cases} x(t + \Delta t) = x(t) + v_c \cdot \Delta t \cdot \cos\left(\theta(t) + \frac{\omega_c \cdot \Delta t}{2}\right) \\ y(t + \Delta t) = y(t) + v_c \cdot \Delta t \cdot \sin\left(\theta(t) + \frac{\omega_c \cdot \Delta t}{2}\right) \\ \theta(t + \Delta t) = \theta(t) + \omega_c \cdot \Delta t \end{cases} \quad (13)$$

Based on the analysis of Equation (13), at any time t , the vehicle's pose can be derived from the velocity set, thus obtaining the displacement, orientation, and motion trajectory information for the time interval Δt . Due to its simplicity and efficiency, the DWA algorithm selects the best path by sampling the vehicle's velocity vectors and evaluating the trajectory, making it a key tool in differential drive vehicle research. To further enhance the algorithm's performance, we will next focus on optimizing the evaluation function of the DWA algorithm.

3.8. Optimization of the Evaluation Function

In the traditional DWA algorithm, the evaluation function scores the trajectory based on factors such as distance to the target point, distance to obstacles, and velocity, considering only the optimal solution in the current environment. However, this evaluation method is prone to getting trapped in local optima in complex environments and does not fully utilize global path information. When encountering U-shaped or concave obstacles, the intelligent vehicle may become stuck, leading to navigation failure.

To address this issue, we propose an optimization method for the evaluation function based on global path guidance. Specifically, the global path generated by the improved RRT* algorithm is introduced, and a global path consistency term, $G_{global}(v, \omega)$, is added to the evaluation function of the DWA algorithm to measure the consistency between the local trajectory and the global path. The formula for the global path consistency term is as follows:

$$\begin{cases} G_{global}(v, \omega) = \frac{1}{1 + d_{global}(v, \omega)} \\ d_{global} = \min_{p \in G_{global}} |p - q_{local_end}| \end{cases} \quad (14)$$

where $d_{global}(v, \omega)$ represents the shortest distance between the end of the local trajectory and the global path; $G_{global}(v, \omega)$ is a global path generated by the RRT* algorithm; and q_{local_end} is the endpoint position of the local trajectory.

By introducing the global path consistency term, the local path is prevented from falling into local optima and deviating from the globally optimal path. After incorporating the global path consistency term, the improved evaluation function of the DWA algorithm is as follows:

$$G(v, \omega) = \alpha \cdot G_{dist}(v, \omega) + \beta \cdot G_{safe}(v, \omega) + \gamma \cdot G_{global}(v, \omega) + \delta \cdot G_{speed}(v, \omega) \quad (15)$$

where $G_{dist}(v, \omega)$ is the distance score to the next target point; $G_{safe}(v, \omega)$ is the safety score; $G_{global}(v, \omega)$ is the global consistency score; $G_{speed}(v, \omega)$ is the velocity score; and α , β , γ , and δ are the weight coefficients for each sub-function.

Next, a 10 m × 10 m two-dimensional map is created to test the obstacle avoidance performance of the traditional DWA algorithm and the improved DWA algorithm. The starting point is set at (0 m, 0 m), and the goal point is at (10 m, 10 m). The black regions represent obstacles, the red region represents the target point, and the green area represents the local region explored by the intelligent vehicle. A U-shaped obstacle is placed in the mandatory path between the start and goal points.

Figure 7 shows that the traditional DWA algorithm causes the intelligent vehicle to slow down and eventually stop at the U-shaped obstacle, resulting in path exploration failure. This is because standard DWA lacks global context and tends to get trapped in local minima when facing narrow passages or concave obstacles. In contrast, Figure 8 shows that the improved DWA algorithm, guided by global path information through the global consistency term, actively avoids the U-shaped obstacle by selecting trajectories that maintain alignment with the global path. This demonstrates that the proposed enhancement not only improves path smoothness but also fundamentally prevents stagnation by enabling proactive obstacle avoidance.

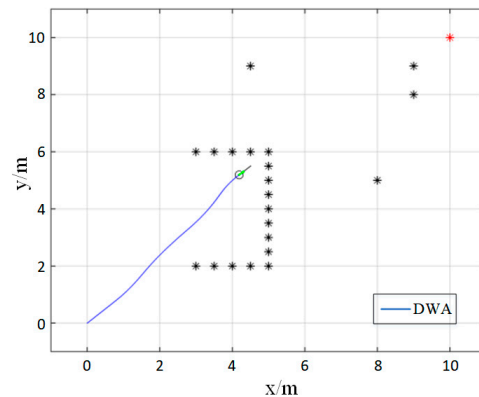


Figure 7. DWA trapped in U-shaped obstacle.

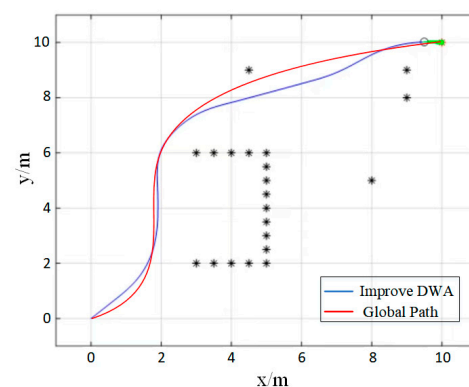


Figure 8. Improved DWA escaping U-shaped obstacle under global path guidance.

4. Single-Target Point Navigation Algorithm

Due to the limitations of global and local path planning, the standard RRT* algorithm can generate a globally optimal path but will struggle with dynamic environments due to inefficient sampling and a lack of obstacle-awareness. Meanwhile, the DWA algorithm excels in real-time obstacle avoidance but may fall into local optima without global guidance. To address these issues, we propose an improved RRT*-DWA integrated navigation framework. The global planner employs a bidirectional RRT* with a constrained sampling region, APF-guided sampling, and adaptive goal bias to enhance efficiency and safety. Specifically, the sampling region is restricted to a rectangle along the line connecting start and goal points, reducing unnecessary exploration. APF potential fields guide samples toward the goal while repelling from obstacles, improving convergence speed. The goal bias probability dynamically adjusts based on local obstacle density: higher in open spaces (0.2) and lower in cluttered areas (0.05), balancing exploration and exploitation. The resulting global path is then smoothed and used as a reference for the DWA-based local planner, which incorporates a cost term for global path following to ensure consistency. This hybrid approach enables the intelligent vehicle to achieve both global optimality and robust dynamic obstacle avoidance in complex environments. The detailed algorithmic flow is illustrated in Appendix C and Figure 9.

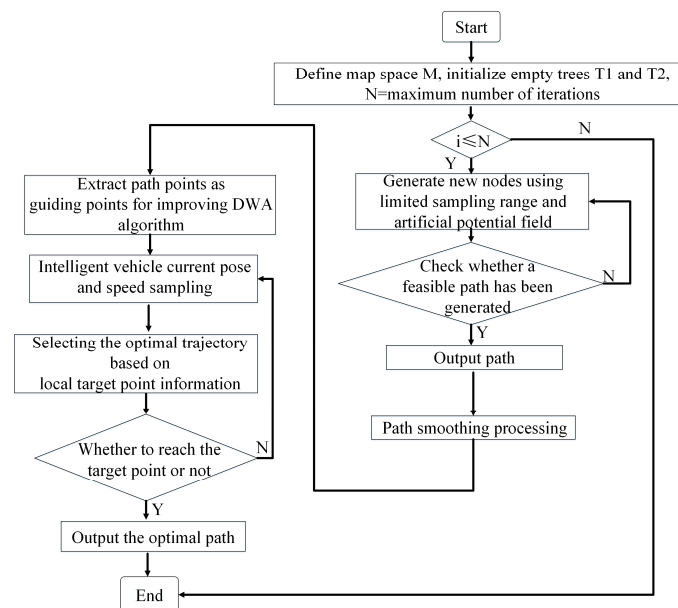


Figure 9. Flowchart of single-target-point navigation algorithm.

5. Simulation Results Analysis

5.1. Comparative Results and Analysis of the Improved RRT* Algorithm

To verify the stability and feasibility of the proposed global path planning algorithm, simulation tests were conducted using MATLAB R2022b. The path smoothing process was removed in the simulation results to better validate the core concept of the algorithm. The proposed algorithm was compared with several advanced algorithms in recent years, including RRT*-Connect, APF-RRT*, and the traditional RRT* algorithm in terms of the average path length, average number of nodes, and average computation time.

Three environments were designed, each with a size of 100 cm × 100 cm and varying obstacle distributions, as shown in Figure 10. The simple environment contains only a few sparse static obstacles. The moderate environment features narrow passages formed by furniture-like static obstacles. The complex environment includes dense U-shaped and concave obstacle configurations that challenge global path planning. Figures 11–13 present the path planning results in the complex, moderate, and simple environments, respectively. In the complex environment (Figure 11), baseline methods such as RRT* and RRT*-Connect generate highly tortuous paths due to unguided sampling, while APF-RRT* suffers from local minima near concave obstacles, often failing to reach the goal. In contrast, the proposed algorithm produces a smooth, near-optimal trajectory by integrating global path consistency into local planning, effectively navigating U-shaped traps. In the moderate environment (Figure 12), the proposed method demonstrates superior corridor navigation—maintaining central alignment and avoiding unnecessary oscillations—whereas other algorithms exhibit zigzagging or excessive detours. Even in the simple environment (Figure 13), our approach achieves the most direct route, confirming its efficiency without over-planning. Figure 14 illustrates the convergence characteristics of the four global path planning algorithms. The proposed method converges significantly faster and achieves a lower final path cost compared with RRT*, RRT*-Connect, and APF-RRT*. This improvement stems from the constrained sampling strategy, which biases random samples toward promising regions along the heuristic direction to the goal, thereby accelerating the discovery of high-quality paths and reducing redundant exploration in obstacle-dense areas. Finally, Table 1 quantitatively summarizes these observations, showing consistent improvements in success rate, path length, and execution time across all scenarios.

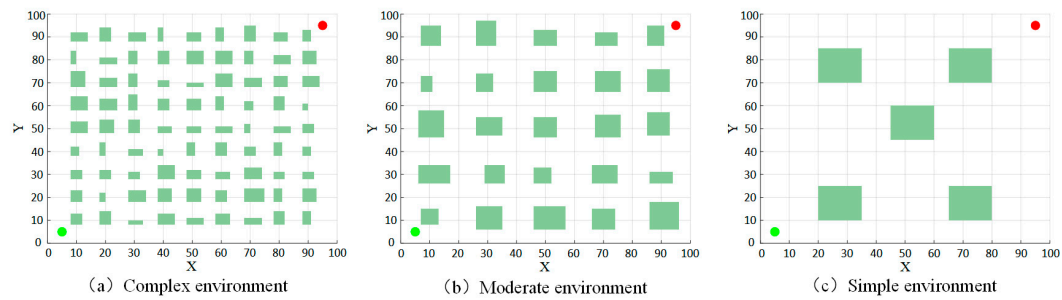


Figure 10. Different simulation environments.

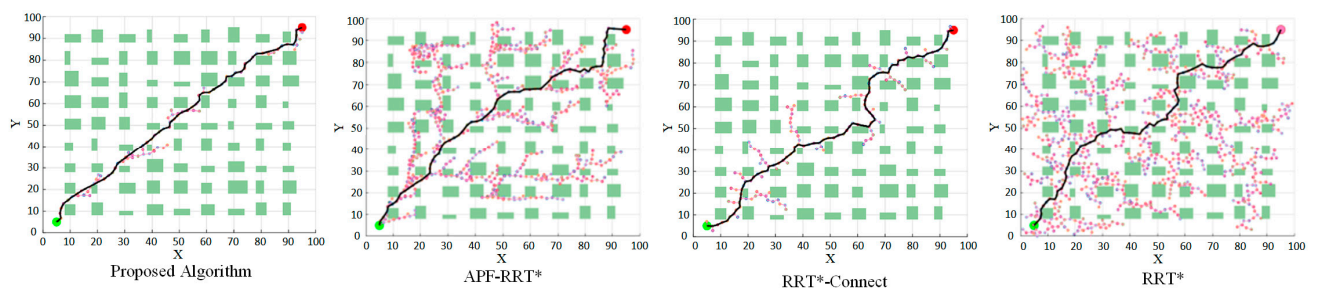


Figure 11. Comparison of complex environment effects.

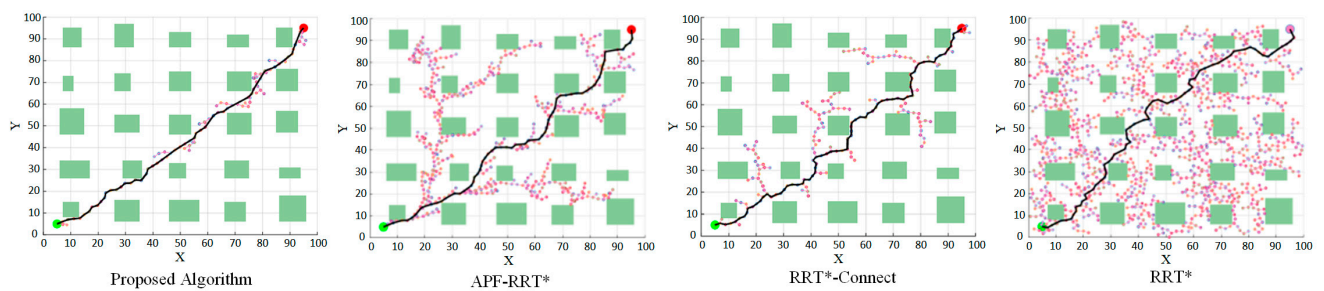


Figure 12. Comparison of moderate environmental effects.

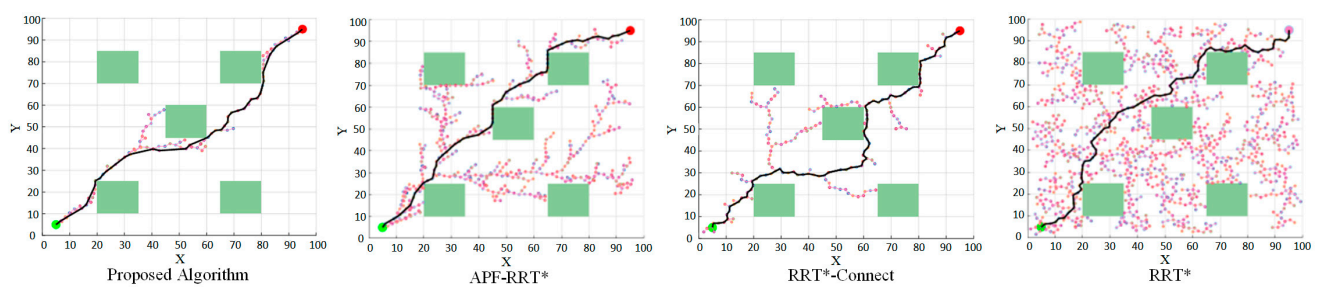


Figure 13. Comparison of simple environmental effects.

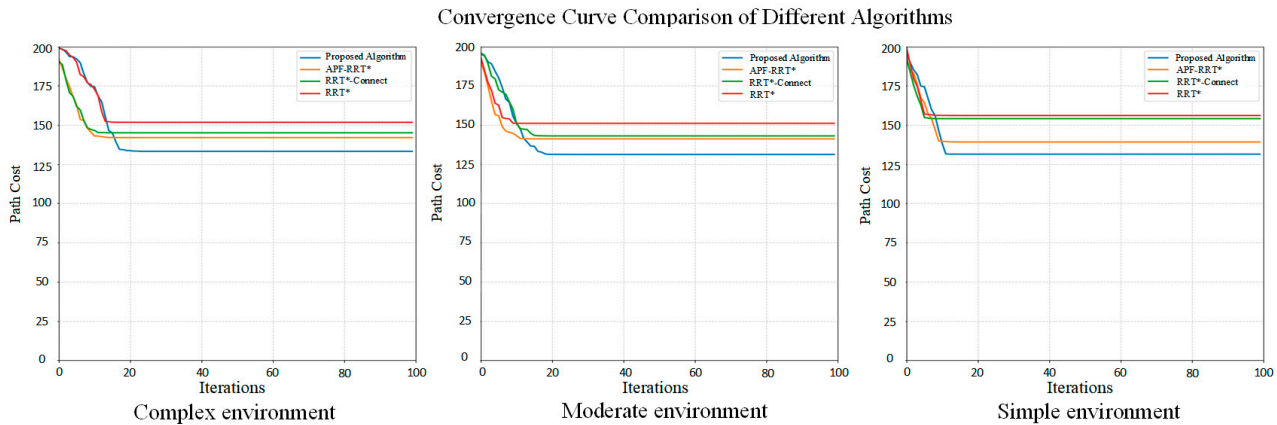


Figure 14. Convergence curve trends of various algorithms in three environments.

In all test scenarios, the starting point was set at (5, 5) and the goal point at (95, 95). Each algorithm was executed 15 times in each of the three environments, with a maximum iteration limit of 500; convergence is defined as the point where the average path length variation over the last 100 iterations falls below 1%. All four algorithms successfully generated a collision-free path from the start to the goal in all test cases.

All simulations were conducted on a standard laptop equipped with an Intel Core i7-1165G7 CPU @ 2.80 GHz (4 cores) and 16 GB RAM, running Windows 11 and MATLAB R2022b. The proposed algorithm was implemented in a single-threaded mode without GPU acceleration, reflecting real-time deployment constraints on embedded systems. In the simulation, the maximum number of iterations for RRT* was set to 500, with a step size of 2.0 m and a collision detection resolution of 0.5 m. The sampling region was constrained to a rectangle centered along the line from start to goal, with a width of 20% of the map size, dynamically adjusted based on local obstacle density. For path smoothing, B-spline interpolation with a smoothing factor, $\lambda=0.5$, was applied to reduce redundant waypoints while preserving path safety. In the DWA-based local planner, the velocity bounds were set to $v \in [0, 3] \text{ m/s}$ and $\omega \in [-1.0, 1.0] \text{ rad/s}$, with a prediction horizon of 1.5 s and a time step of 0.1 s. The cost weights were configured as $\omega_{\text{obs}} = 1.0$, $\omega_{\text{vel}} = 0.2$, $\omega_{\text{goal}} = 0.5$, and $\omega_{\text{global}} = 0.8$ to balance obstacle avoidance, smoothness, goal-reaching, and global path following. These parameters were consistent across all algorithms for fair comparison. The average computation time per run was measured over 15 independent trials, with peak memory usage below 50 MB, demonstrating that the algorithm achieves real-time performance on commodity hardware.

The simulation results show that the path generated by the proposed algorithm is not only closer to the optimal solution but also reduces the selection of unnecessary nodes in non-essential areas. In contrast, the RRT*-Connect algorithm adopts a bidirectional exploration strategy, which significantly improves exploration speed compared with RRT*. However, its aimless exploration also generates some unnecessary nodes, leading to lower path quality. The APF-RRT* algorithm utilizes information from obstacles, the start point, and the goal point to guide the exploration tree to avoid obstacles and move toward the goal. Although this approach significantly reduces the generation of redundant nodes, the repulsive forces from clusters of obstacles may cause the algorithm to generate nodes deviating from the ideal path or become trapped in densely obstructed areas, thereby affecting its convergence speed and path quality. Similarly, due to its aimless exploration, the RRT* algorithm generates many unnecessary nodes, resulting in poor path quality.

According to Table 2, all algorithms successfully generated collision-free paths in every trial across all test environments, achieving a 100% success rate. In complex environments, the proposed algorithm outperforms the others by achieving shorter search times and shorter path lengths. Compared with the RRT* algorithm, the proposed algorithm reduces the average number of nodes by 88.7%, decreases the average path length by 12.2%, and shortens the average computation time by 95.3%.

Table 2. Comparison of various datapoints in simulation results.

Environment	Algorithm	Avg. Path Length (cm)	Avg Number of Nodes	Times (s)
Complex environment	Proposed Algorithm	133.06	98.00	1.56
	APF-RRT*	141.94	828.00	19.73
	RRT*-Connect	145.33	289.00	8.94
	RRT*	151.47	868.00	33.15
Moderate environment	Proposed Algorithm	131.58	106.00	1.48
	APF-RRT*	141.23	597.00	12.84
	RRT*-Connect	143.58	308.00	9.95
	RRT*	151.30	1119.00	31.02
Simple environment	Proposed Algorithm	131.42	112.00	1.94
	APF-RRT*	139.79	622.00	10.96
	RRT*-Connect	154.83	268.00	8.96
	RRT*	156.39	1047.00	25.76

In moderately complex environments, the proposed algorithm reduces the average path length by 13.0%, decreases the average number of nodes by 90.5%, and shortens the average computation time by 95.2% compared with the RRT* algorithm.

To further compare the performance of the improved algorithm, 100 simulation results were conducted in three different environments: simple, moderate, and complex. The average path length, average execution time, average number of iterations, and average number of nodes were recorded. A comparison chart of simulation data in three environments is shown in Figure 15.

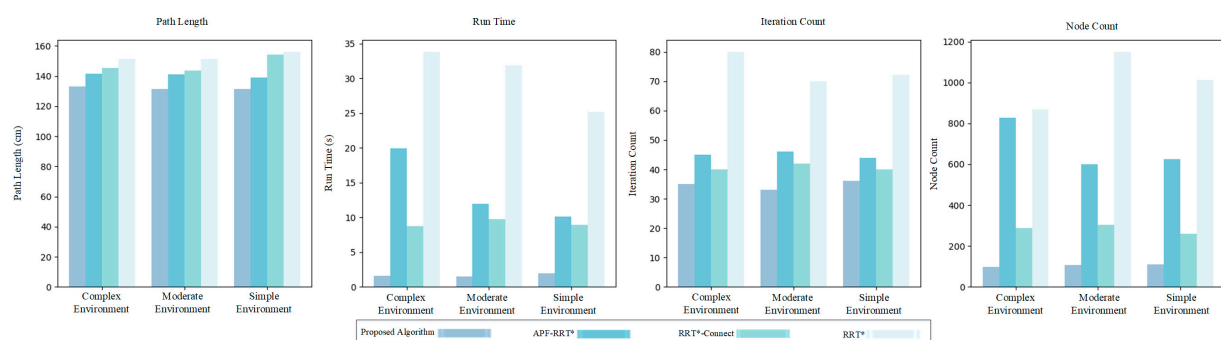


Figure 15. Comparison chart of simulation data in three environments.

The proposed algorithm achieves substantial improvements over the traditional RRT*, APF-RRT*, and RRT*-Connect algorithms in all three environments. Specifically, it reduces the average path length by 12.1–15.9% while dramatically cutting the number of generated nodes by over 88% and planning runtime by more than 92% (see Table 1). These gains stem from the sampling bias toward goal-directed regions, which minimizes exploration in irrelevant or obstacle-dense areas. Consequently, the planner constructs high-quality paths with far fewer samples, significantly lowering computational overhead. This efficiency enables real-time replanning on resource-constrained platforms—such as our

differential-drive robot—and directly contributes to smoother, faster navigation in practice.

To quantify the contribution of each improvement module, we gradually introduce improvement strategies and test performance (environment complexity: moderate).

Table 3 shows that bidirectional search has the most significant improvement in convergence speed (67.9%), while APF fusion contributes the most to path optimization (11.8% length reduction). The dynamic bias strategy further reduces computation time by 15%.

Table 3. Ablation study of RRT* improvements.

Configuration	Avg. Length (cm)	Δ Length	Times (s)	Δ Time	Nodes
Baseline RRT*	151.30 $\pm$ 1.2	-	31.02 $\pm$ 2.1	-	1119 $\pm$ 42
+Bidirectional Search	143.58 $\pm$ 0.9	-5.1%	9.95 $\pm$ 0.7	-67.9%	308 $\pm$ 25
+Constrained Sampling	138.47 $\pm$ 0.7	-8.5%	4.83 $\pm$ 0.4	-84.4%	195 $\pm$ 18
+APF Integration	134.92 $\pm$ 0.6	-10.8%	2.76 $\pm$ 0.3	-91.1%	127 $\pm$ 15
Full Proposed Method	131.58 $\pm$ 0.5	-13.0%	1.48 $\pm$ 0.2	-95.2%	106 $\pm$ 12

5.2. Comparative Results and Analysis of the Improved RRT*-DWA Algorithm

To verify the stability and feasibility of the proposed local path planning algorithm, simulation tests were conducted in a 10 m $\times$ 10 m two-dimensional environment using the MATLAB R2022b platform. The improved RRT*-DWA algorithm proposed in this paper was compared with Dijkstra-DWA and A*-DWA in terms of path length, average speed, and runtime (Table 4). The starting point for all scenarios was set at (0, 0), and the goal point was (10, 10). The path planning results of the three different algorithms are illustrated in Figure 16, where the blue path represents the actual executed path, and the orange path represents the global planned path. All three frameworks share the same local planner (standard DWA) with identical parameters and operate on the same 10 m $\times$ 10 m occupancy grid map. The only difference lies in the global path planner: our method uses the improved RRT*, while the baselines employ Dijkstra and A*.

Table 4. Comparison of various simulation results data.

Algorithm	Average Path Length (m)	Average Speed (m/s)	Times (s)
Improved RRT*-DWA	15.21	2.45	6.2
Dijkstra-DWA	15.96	1.39	11.5
A*-DWA	15.88	1.37	11.6

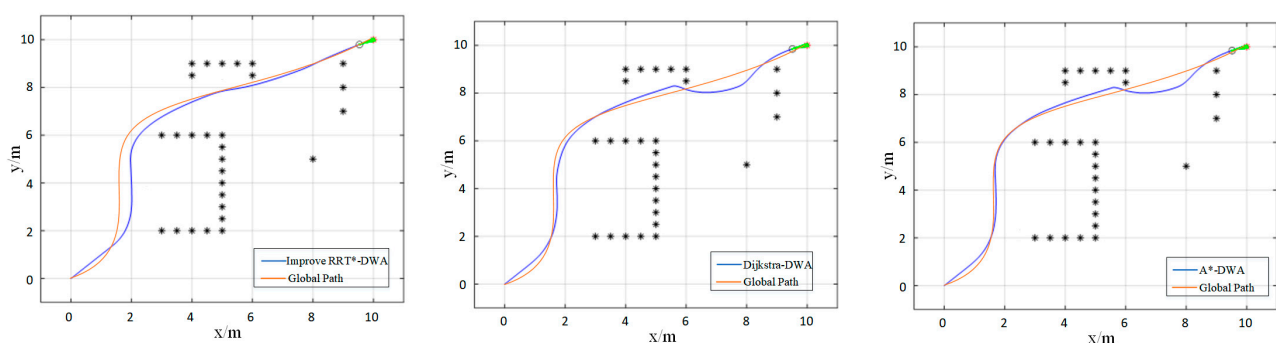


Figure 16. Planning results for the three algorithms.

In the comparison of the results for the three algorithms in Figure 16, two U-shaped obstacles are present in the map—one large and one small. Although both the A-DWA* and Dijkstra-DWA algorithms successfully avoid the first large U-shaped obstacle, their

global paths pass too close to the second smaller U-shaped obstacle. Consequently, the paths generated by A-DWA* and Dijkstra-DWA enter a small section of the U-shaped obstacle area, increasing the risk of collision. This situation could lead to path planning failure or require extra time to adjust the path.

Table 4 shows that in terms of the path length, the improved RRT*-DWA algorithm reduces the length by 4.7% and 4.2% compared with Dijkstra-DWA and A*-DWA, respectively. Regarding average speed, the improved RRT*-DWA algorithm achieves increases of 76% and 79% over Dijkstra-DWA and A*-DWA, respectively. Furthermore, in runtime, the improved RRT*-DWA algorithm reduces execution time by 46% and 47%, respectively.

Overall, the results demonstrate that the improved RRT*-DWA algorithm not only enables safer and more effective avoidance of all U-shaped obstacles but also produces shorter and smoother planned paths.

To verify the dynamic obstacle avoidance capabilities of the improved RRT*-DWA algorithm, a moving obstacle was added to the map in Figure 16. This obstacle moves at a constant speed from the starting point (7.5, 2) to the endpoint (7.5, 10).

The obstacle avoidance sequence in Figure 17 (time steps a–f) demonstrates the reactive capabilities of the enhanced local planner in dynamic scenarios. At step d, upon detecting an approaching moving obstacle, the planner proactively adjusts the robot's heading—not through abrupt braking or oscillation, but via a smooth trajectory deviation that maintains forward progress. By step e, the vehicle has safely overtaken the obstacle and begins realigning with the global reference path, confirming that the local planner preserves long-term navigation intent rather than getting trapped in short-term reactions. This behavior highlights the effectiveness of integrating goal-directed guidance into local decision-making, enabling both safety and efficiency in human–robot shared spaces.

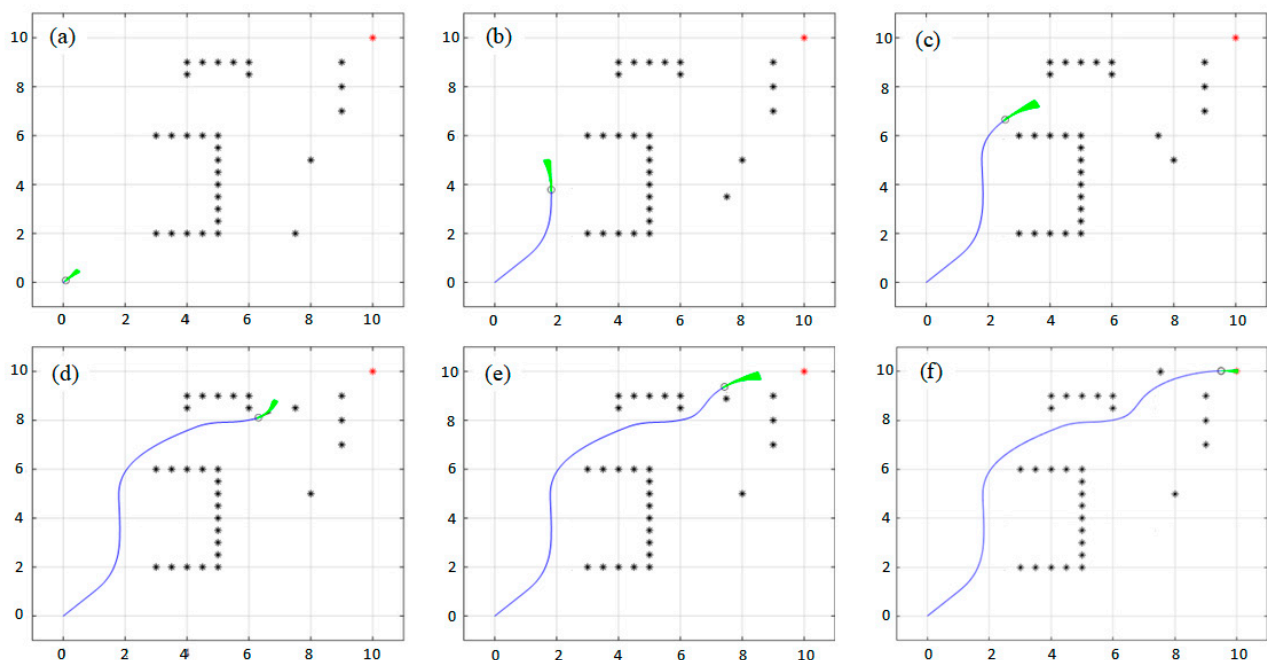


Figure 17. Dynamic obstacle avoidance process of improved RRT*-DWA algorithm.

The simulation results demonstrate that the improved RRT*-DWA algorithm effectively navigates from the start point to the goal point with stability, regardless of whether the obstacles are static or dynamic.

6. Physical Experiment and Discussion

To verify the dynamic obstacle avoidance performance of the integrated navigation algorithm in a real-world environment, experiments were conducted using a two-wheel differential drive WHEELTEC ROS robot. In the experiments, a 5 m × 5 m experimental field was set up, resembling the U-shaped obstacle map layout mentioned earlier. The specific structure of the robot and the experimental field are shown in Figure 18a and Figure 18b, respectively. The robot is equipped with an OpenCTR B20 motion controller based on STM32F103, an RPLIDAR A2 (6 m range, 0.45° angular resolution, 10 Hz scan rate), and two 24 V DC brushed motors with encoders, capable of reaching a maximum linear speed of 0.6 m/s and angular speed of 2.0 rad/s. The onboard computer is a Raspberry Pi 4 running Ubuntu 18.04 and ROS Melodic. Localization is achieved via AMCL with an accuracy of approximately ±0.05 m. During testing, the robot demonstrated an average path deviation of 0.08 m (maximum 0.15 m) from the global path, maintained a real-time planning frequency of 10 Hz, and achieved a 100% obstacle avoidance success rate across 10 trials with dynamic obstacles moving at 0.3–0.5 m/s. The physical experiments were conducted in the robotics laboratory at Tianjin Sino-German University of Applied Sciences, in an indoor area of approximately 5 m × 5 m. The environment was configured with multiple static traffic cones to form cluttered and narrow passages. To simulate a predictable moving obstacle, one cone was tethered with a thin string and pulled along a straight line from (3.25, 1) to (3.25, 5) by an operator positioned at the destination point. This setup ensures a consistent and repeatable dynamic intrusion during robot navigation, mimicking scenarios such as pedestrians or carts moving along known pathways in service environments.

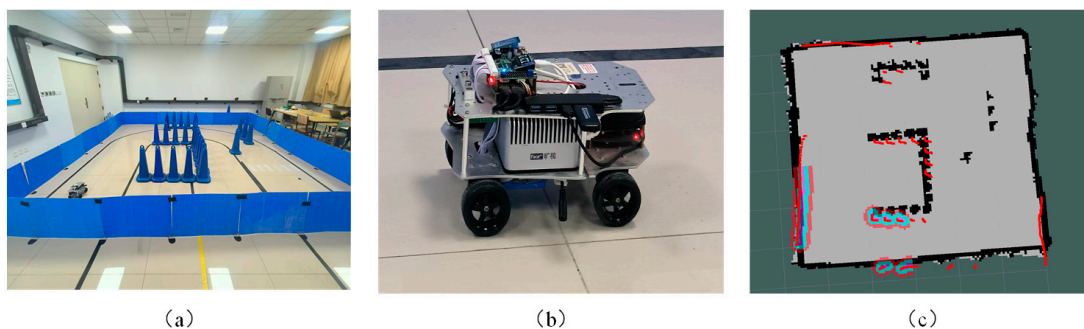


Figure 18. Physical experimental environment and equipment.

The experimental system adopted a master–slave architecture, with the personal computer acting as the master and the intelligent vehicle's upper computer as the slave. The two devices were connected via the vehicle's built-in Wi-Fi module to ensure stable and real-time data transmission. The master terminal activated the keyboard control node to enable remote control of the intelligent vehicle. Subsequently, the slave terminal started the mapping node based on the Gmapping algorithm, which controlled the vehicle for environmental perception and grid map construction. The constructed map is shown in Figure 18c. Our algorithm implementation is based on a set of thoroughly validated core parameter configurations that demonstrate excellent stability and adaptability in typical indoor environments. For the RRT* algorithm component, we employ a base connection threshold (d_{base}) of 1.5 m (recommended range 1.0–2.5 m) and an obstacle density threshold (δ range) of [0.3, 0.35], combined with a directional preference coefficient (λ_d) of 0.6 to balance exploration efficiency and path quality. The DWA local planner utilizes a global path consistency weight (γ) of 0.4 and a safety distance weight (ω_3) of 0.8 to ensure navigation reliability and safety. For researchers interested in verifying the algorithm's core

functionality, integration with ROS systems can be achieved through MATLAB's ROS Toolbox: we first implement the RRT* core logic in MATLAB and then establish ROS network connectivity to publish path topics, with the ROS side ultimately receiving and executing the path planning results. In specialized environment applications, we recommend targeted adjustments based on scenario characteristics: narrow spaces may benefit from increasing the directional preference coefficient to 0.7–0.9, obstacle-dense environments ($\delta > 0.4$) may require expanded sampling region width, and dynamic environments could see improved performance by raising the global path weight to 0.5–0.6. These parameter combinations and adjustment strategies have proven effective in various indoor environment tests, with detailed experimental data available in the paper's comprehensive analysis.

6.1. Global Path Planning Comparison Experiment

Experiments were conducted by navigating an intelligent vehicle to compare the global path planning performance of the proposed algorithm, APF-RRT*, and RRT*-Connect in a real-world environment. Figure 19 shows that both APF-RRT* and RRT*-Connect can plan a safe, collision-free path from the start point to the goal. However, the global paths planned by these algorithms contain too many sharp turns, resulting in worse path quality. In contrast, the global path planned by the proposed algorithm is significantly simplified, with all segments maintaining a regular straight line, except at the turns. This leads to a notable improvement in path quality. The experimental results are compared in Table 5.

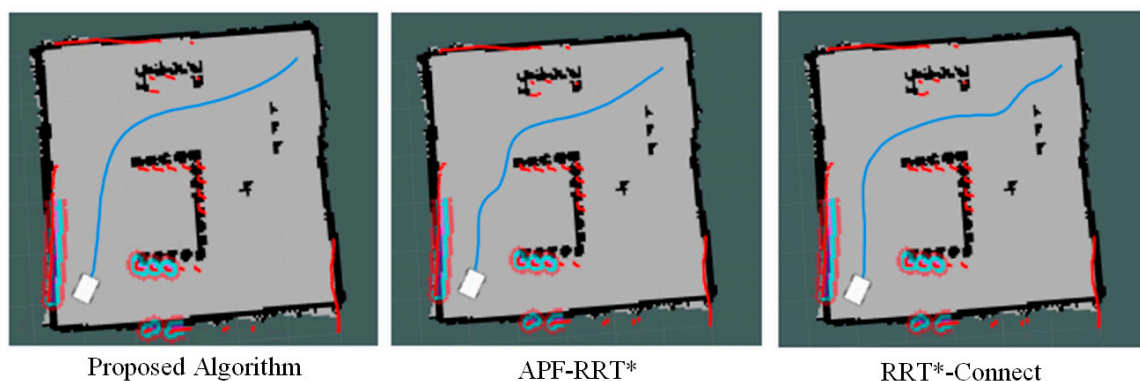


Figure 19. Three different algorithms for global navigation experiments.

Table 5. Comparison of experimental results.

Algorithm	Path Length (m)	Times (s)
Proposed Algorithm	7.61	10.6
APF-RRT*	8.35	14.5
RRT*-Connect	8.47	15.2

Compared with APF-RRT* and RRT*-Connect, the proposed global path planner achieves a shorter path length (by 8.9% and 10.2%, respectively) and a significantly lower planning time (reduced by 26.9% and 30.3%) (Table 5). These improvements result from the constrained sampling mechanism, which focuses exploration along heuristic directions toward the goal and avoids redundant node expansion in low-potential regions. The reduced runtime is particularly beneficial for real-world deployment, as it enables more frequent replanning cycles under dynamic conditions while maintaining low computational load—enhancing both responsiveness and system stability.

6.2. Physical Experiment and Comparison of Improved RRT*-DWA Algorithm

To verify the practical effectiveness of the improved RRT*-DWA algorithm, an experimental field with dynamic obstacles was set up, resembling the layout in Figure 17. Figure 20 shows the entire experimental process at nine time points, from a to i. At time point e, the intelligent vehicle detects a dynamic obstacle ahead. At time point f, the vehicle adjusts its path to prepare for avoiding the dynamic obstacle, and by time point h, it successfully avoids the obstacle. The experimental results are compared in Table 6.

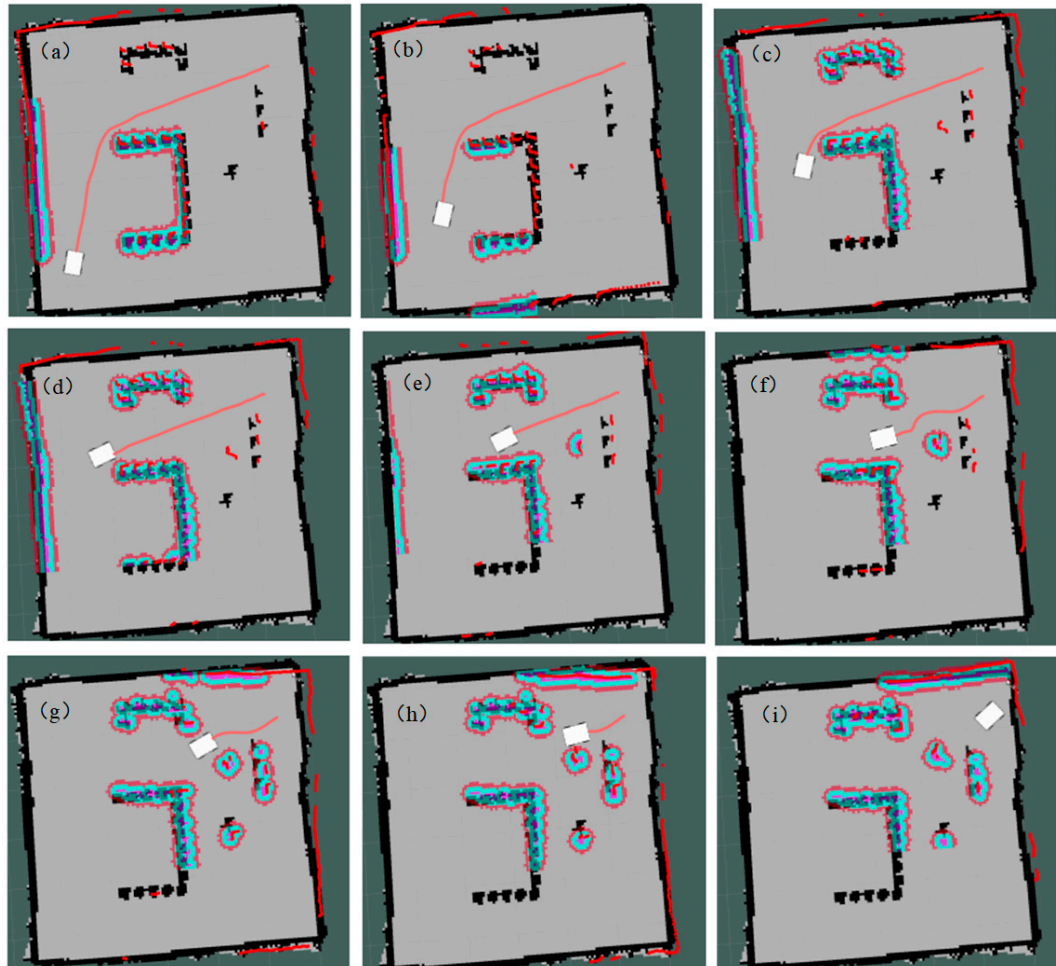


Figure 20. Dynamic obstacle avoidance process of improved RRT*-DWA algorithm in a physical experiment.

Table 6. Comparison of experimental data.

Algorithm	Path Length (m)	Times (s)
Improved RRT*-DWA	7.75	11.2
Dijkstra-DWA	8.16	16.5
A*-DWA	8.19	17.1

The proposed improved RRT*-DWA algorithm outperforms both Dijkstra-DWA and A*-DWA in terms of path efficiency and computational speed (Table 6): it reduces path length by 5.0% and 5.4% and planning runtime by 32.1% and 34.5%, respectively. This advantage arises from the continuous-space sampling of RRT*, which generates smoother, curvature-feasible global paths compared with the grid-constrained routes of Dijkstra and A*. Consequently, the local DWA planner requires fewer corrective maneuvers, leading

to more consistent velocity profiles and reduced oscillation during navigation—key indicators of operational stability. The tighter coupling between global and local planning thus enables more efficient and reliable autonomous navigation in cluttered indoor environments.

6.3. Analysis and Discussion

The physical experiments confirm the effectiveness of the improved RRT*-DWA algorithm in real-world navigation scenarios. By guiding the sampling process of RRT* toward the goal direction, the global planner generates smoother and shorter paths than Dijkstra-DWA and A*-DWA, which rely on discrete grid representations. This continuous-space planning better aligns with the kinematic characteristics of the two-wheel differential-drive WHEELTEC robot, reducing unnecessary heading adjustments and velocity fluctuations during execution—evident in the stable trajectory shown in Figure 19.

Moreover, the successful avoidance of the tethered moving cone (pulled from (3.25, 1) to (3.25, 5)) demonstrates that the local DWA planner can respond to dynamic obstacles while maintaining adherence to the global path. The robot consistently returned to its intended route after overtaking the obstacle, as observed in Figure 20, achieving a 100% success rate across all trials. This behavior highlights the robust coupling between global guidance and local reactivity in our integrated framework.

Despite these advantages, the current implementation has limitations. It assumes a pre-built static map and predictable obstacle motion, which may not hold in highly dynamic or unknown environments such as crowded public areas. Additionally, while the system maintains a real-time planning frequency of 10 Hz on the Raspberry Pi 4, computational load could increase significantly in larger or more complex maps.

Future work will focus on extending the framework to support online map updates through SLAM integration, incorporating simple motion prediction for faster-moving obstacles, and exploring adaptive parameter tuning for DWA based on local scene complexity. These enhancements aim to improve robustness and generalization in more diverse real-world applications.

7. Conclusions and Future Works

This paper addressed the challenge of inefficient and unstable navigation in cluttered indoor environments by proposing an improved RRT*-DWA integrated framework. The key innovation lies in guiding the RRT* sampling process toward the goal direction while maintaining asymptotic optimality, which enables the generation of smoother, shorter, and kinematically feasible global paths. When tightly coupled with the DWA local planner, this approach significantly enhances navigation efficiency and robustness—particularly in scenarios involving dynamic obstacles—as validated through both simulation and physical experiments on a WHEELTEC differential-drive robot.

While the current system demonstrates strong performance under known static maps and predictable moving obstacles, future work will explore online map updating via SLAM, motion prediction for erratic obstacles, and extension to multi-robot coordination to broaden its applicability in real-world service robotics.

Author Contributions: Methodology, Y.Z.; Software, X.L., Z.G., Z.L. and W.W.; Formal analysis, Z.L. and W.W.; Data curation, X.L.; Writing—original draft, Y.Z.; Funding acquisition, Z.G. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Tianjin Science and Technology Leading Enterprise Project (Project Number: 23YDPYGX00180), Enterprise Technology Commissioner Project (Project Number: 25YDTPJC00260), Tianjin Jinnan District unveiling and leading project (Project Number: H2025-095).

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

Conflicts of Interest: Zhijie Li and Weiqiang Wang are employees of Tianjin Deke Intelligent Control Co., Ltd. The authors declare that this study was supported by the software code of the company. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Appendix A

Algorithm A1. Algorithm RRT*.

```

1. Input:  $q_{start}$ ,  $q_{goal}$ , Map,  $N_{max}$ 
2. Output: Global path
3. Initialize tree T with  $q_{start}$ 
4. for i = 1 to MaxIterations do
5.    $q_{rand}$  = RandomSample( Map );
6.    $q_{near}$  = NearestNeighbor( T,  $q_{rand}$  );
7.    $q_{new}$  = Steer(  $q_{near}$ ,  $q_{rand}$  );
8.   if CollisionFree(  $q_{near}$ ,  $q_{new}$  ) then
9.      $q_{naer\_nodes}$  = NearNodes( T,  $q_{new}$ , radius );
10.     $q_{min}$  = ChooseParent(  $q_{naer\_nodes}$ ,  $q_{new}$  );
11.    AddNode( T,  $q_{new}$ ,  $q_{min}$  );
12.    Rewire( T,  $q_{new}$ ,  $q_{naer\_nodes}$  );
13.    if Distance(  $q_{new}$ ,  $q_{goal}$  ) < threshoud then
14.      return Path( T,  $q_{start}$ ,  $q_{goal}$  );
15.    end if;
16.  end if;
17. end for
18. return NoPathFound

```

Appendix B

Algorithm A2. Determination steps for rectangular sampling area.

```

1. Input:  $q_{start}$  and  $q_{goal}$  and obstacle_map
2.  $[\delta_{min}, \delta_{max}] = [0.3, 0.35]$ ;
3. Output: W
4.  $L = \sqrt{(x.start - x.goal)^2 + (y.start - y.goal)^2}$  ;
5. W = 1.0,  $\Delta W = 0.1$ ;
6. while True:
7.   Area = L * W ;
8.   obstacle_count = 0;
9.   for k = 1 to N:

```

```

10.      p = RandomPointInRegion( );
11.  if CheckCollision(p,obstacle_map) then
12.      obstacle_count += 1;
13.   $\delta = \text{obstacle\_count}/N$ ;
14.  if  $\delta_{\min} \leq \delta \leq \delta_{\max}$ 
15.      break;
16.  W = W +  $\Delta W$  ;
17. Return W ;

```

Appendix C

Algorithm A3. Improved RRT*-DWA path planning.

```

1. Input:  $q_{start}$  and  $q_{goal}$  and obstacle_map M and  $N_{\max}$ 
2. Output: P(smoothed global path) and traj(executed local trajectory)
3: Create tree  $T_{start}$  rooted at  $q_{start}$  and tree  $T_{goal}$  rooted at  $q_{goal}$  ;
4:  Center_line = line from  $q_{start}$  to  $q_{goal}$  ;
5:  R = rectangle along center_line, width = 30% of M width; //Width can be adjusted based on local obstacle density
6:  for i = 1 to  $N_{\max}$  do //Dynamic goal bias based on obstacle density
7:       $\Delta = \text{obstacle\_density\_in\_region}(R)$ ;
8:      if  $\Delta < 0.2$  then
9:           $p_{goal} = 0.2$ ; //In open space, bias more toward goal
10:     else;
11:          $p_{goal} = 0.05$ ; //In cluttered space, explore more
12:     end if;
13:     if rand() <  $p_{goal}$  then //Sample with APF guidance within R
14:          $q_{rand} = q_{goal}$  ;
15:     else; //Sample within R, biased by APF (attractive to goal, repulsive to obstacles)
16:          $q_{rand} = \text{sample\_from}(R, \text{using\_APF\_potential})$ ;
17:     end if;
18:     if i % 2 == 0 then //Alternate tree expansion
19:          $T_{active} = T_{start}$  ,  $T_{target} = T_{goal}$  ;
20:     else;
21:          $T_{active} = T_{goal}$  ,  $T_{target} = T_{start}$  ;
22:     end if;
23:      $q_{near} = \text{nearest\_node}(T_{active}, q_{rand})$ ;
24:      $q_{new} = \text{extend}(q_{near}, q_{rand})$ ;
25:     if is_collision_free( $q_{near}$ ,  $q_{new}$ , M) then
26:         add  $q_{new}$  to  $T_{active}$  ;

```

```

27:         rewire_neighbors( $T_{active}$ ,  $q_{new}$ ); //Optimize path cost
28:         if distance( $q_{new}$ , any node in  $T_{active}$ ) < 0.5 then //Check if trees can be con-
            nected
29:             connect_trees( $T_{start}$ ,  $T_{goal}$ );
30:             break; //Exit loop
31:         end if;
32:     end if;
33: end for;
34: P_raw = extract_path( $T_{start}$ ,  $T_{goal}$ ); //Extract and smooth global path
35: P = smooth_path(P_raw); //Remove redundant waypoints
36: current_pose = x_start; //Local planning: DWA with Global Path Following
37: traj = empty list;
38: while distance(current_pose,  $q_{goal}$ ) > 0.3 do
39:     candidate_trajs = DWA.generate_trajectories(current_pose);
40:     best_traj = select_lowest_cost(candidate_trajs, P, M);
41:     append best_traj to traj; //Cost includes: obstacle avoidance, speed, goal reaching,
        and global path following
42:     current_pose = final_pose(best_traj);
43:     if deviation_from_path(current_pose, P) > 0.5 or new_obstacle_detected then
        //Replan global path
44:         goto line 2; //Trigger global replanning
45:     end if;
46: end while;
47: Return P, traj;

```

Reference

1. Nobari, O.; Nobari, A.; Eslamipoor, R. Optimal point-to-point path planning of manipulator by using vibration dam** optimization algorithm and game theory method. *J. Test. Eval.* **2019**, *47*, 2867–2888.
2. Budhia, M.; Boys, J.T.; Covic, G.A.; Huang, C.Y. Development of a single-sided flux magnetic coupler for electric vehicle IPT charging systems. *IEEE Trans. Ind. Electron.* **2011**, *60*, 318–328.
3. Wang, H.; Zhuo, S.; Gao, F.; Breaz, E.; Gaillard, A.; Cheng, K.W.E. An Integrated Design of Cost-effective Bipolar Hexagonal Coil and Active Disturbance Rejection Control for Wireless Power Transfer. In *Proceedings of the 2022 IEEE Transportation Electrification Conference; Expo (ITEC)*; IEEE: Piscataway, NJ, USA, 2022; pp. 617–622.
4. Sheng, X.; Yang, J.; You, L.; Li, J.; Wang, R. GOA-ACO: A goose optimized ant colony algorithm for the automated guided vehicle path planning. *Alex. Eng. J.* **2025**, *130*, 724–737. <https://doi.org/10.1016/J.AEJ.2025.09.036>.
5. Zhao, R.; Zhang, H.; Liu, C.; Fu, L.; Kang, J.; Shi, Y.; Jiang, C. Ant colony optimization using eight-neighborhood search with dual pheromone updating strategy and continuous image edge generation strategy for license plate recognition. *Clust. Comput.* **2025**, *28*, 922. <https://doi.org/10.1007/S10586-025-05638-W>.
6. Liu, H.; Sun, J.; Wang, H.; Cheng, K.W.E. Comprehensive analysis of adaptive soft actor-critic reinforcement learning-based control framework for autonomous driving in varied scenarios. *IEEE Trans. Transp. Electrification* **2024**, *11*, 3667–3679.
7. Shen, L.; Chi, J.; Hao, M.; Zhao, Z.; Zhang, X.; Chen, D.; Zhou, R.; Bian, H.; Jiang, J. Ant colony algorithm guided synthesis of flame retardants for enhanced polylactic acid flame retardancy. *Eur. Polym. J.* **2025**, *237*, 114192. <https://doi.org/10.1016/J.EUR-POLYMJ.2025.114192>.
8. Yu, Z.; Liu, S.; Song, M.; Fu, X.; Hao, Z. Intelligent Environment-aware Path Planning Based on Ant Colony Optimization for Unmanned Sailboats in Complex Marine Environments. *J. Phys. Conf. Ser.* **2025**, *3108*, 012026. <https://doi.org/10.1088/1742-6596/3108/1/012026>.

9. Yang, H.; Liu, C.; Ren, J.; Zeng, X. Research on path planning based on improved double layer (ordinary and elite) ant colony algorithm. *Eng. Res. Express* **2025**, *7*, 035264. <https://doi.org/10.1088/2631-8695/ADF9C2>.
10. Li, X.; Huang, F.; Sun, H.; He, J.; Javidan, S.M.; Ampatzidis, Y.; Zhang, Z. A bio-inspired framework for apple leaf disease detection: Integrating lesion localization, ant colony optimization, and machine learning. *Smart Agric. Technol.* **2025**, *12*, 101141. <https://doi.org/10.1016/J.ATECH.2025.101141>.
11. Elkawagy, M.; Elgendy, I.A.; Muthanna, A.; Alkanhel, R.I.; Elbeh, H. Enhancing Hierarchical Task Network Planning through Ant Colony Optimization in Refinement Process. *Comput. Mater. Continua* **2025**, *84*, 393–415. <https://doi.org/10.32604/CMC.2025.063766>.
12. Zhang, T.; He, Q.; Zhao, W.; Wei, M. Sustainable closed-loop supply chain network planning considering price competition using particle chaotic ant colony algorithm. *Sci. Rep.* **2025**, *15*, 17964. <https://doi.org/10.1038/S41598-025-02656-X>.
13. Achaa-Amankwaa, P.; Trautwein, T.; Lenhard, W.; Schroeders, U. Balancing Between Categorical and Dimensional Assessment in Short-Scale Construction Using Ant Colony Optimization. *Eur. J. Psychol. Assess.* **2025**. <https://doi.org/10.1027/1015-5759/A000892>.
14. Huang, Y.; Li, H.; Dai, Y.; Lu, G.; Duan, M. A 3D Path Planning Algorithm for UAVs Based on an Improved Artificial Potential Field and Bidirectional RRT*. *Drones* **2024**, *8*, 760.
15. Ye, B.; Liu, C.; Han, F.; Hu, X.; Chen, L. Obstacle avoidance motion planning for space redundant manipulator based on improved RRT algorithm. *J. Phys. Conf. Ser.* **2024**, *2764*, 012052.
16. Wang, Z.; Li, G. Research on path planning algorithm of driverless ferry vehicles combining improved A* and DWA. *Sensors* **2024**, *24*, 4041.
17. Wang, H.; Ma, X.; Zhu, L. Obstacle avoidance path planning of mobile robot based on improved DWA. *J. Phys. Conf. Ser.* **2022**, *2383*, 012098.
18. Suwoyo, H.; Burhanudin, A.; Tian, Y.; Andika, J. Problem solving path planning and path tracking in a 3 DOF hexapod robot using the RRT* algorithm with path optimization and Pose-to-Pose. *Sinergi Indones* **2024**, *28*, 265–276.
19. Ge, Y.; Eidsvik, J.; Olaisen, A.J.H. RRT*-enhanced long-horizon path planning for AUV adaptive sampling using a cost valley. *Knowl.-Based Syst.* **2025**, *315*, 113261.
20. Oh, T.; Won, Y.J.; Lee, S. TA-RRT*: Adaptive Sampling-Based Path Planning Using Terrain Analysis. *Appl. Sci.* **2025**, *15*, 2287.
21. Liu, N.; Hu, Z.; Wei, M.; Guo, P.; Zhang, S.; Zhang, A. Improved A* algorithm incorporating RRT* thought: A path planning algorithm for AGV in digitalised workshops. *Comput. Oper. Res.* **2025**, *177*, 106993.
22. Zhang, X.; Wang, P.; Guo, Y.; Han, Q.; Zhang, K. Path Planning Algorithm for Manipulators in Complex Scenes Based on Improved RRT*. *Sensors* **2025**, *25*, 328.
23. Feng, Z.; Zhou, L.; Qi, J.; Hong, S. DBVS-APF-RRT*: A global path planning algorithm with ultra-high speed generation of initial paths and high optimal path quality. *Expert Syst. Appl.* **2024**, *249*, 123571.
24. Kun, W.; Guohui, Z.; Dunke, L.; Bo, H.; Xiaobin, L. A mobile robot path planning algorithm based on an improved asymptotic optimal bidirectional fast expanding random tree. *Comput. Appl.* **2019**, *39*, 1312–1317.
25. Dan, Y.; Huang, B.; Feng, G. Research on Global Path Planning for Unmanned Vehicles with Improved RRT * Algorithm. *Comput. Eng. Appl.* **2025**, *61*, 326–335.
26. Guiseng, Y.; Jie, Y. Improved Informed RRT * path planning algorithm based on seahorse optimization. *Mech. Transm.* **2025**, *49*, 93–100. <https://doi.org/10.16578/j.issn.1004.2539.2025.02.012>.
27. Park, H.M.; Cho, S.W.; Seo, K.M. Hybrid path planning framework to integrate improved A*-DWA algorithms for enhancing path safety and efficiency. *Appl. Ocean Res.* **2025**, *157*, 104497.
28. He, Q.; Wang, Z.; Li, K.; Zhang, Y.; Li, M. Research on autonomous navigation of mobile robots based on IA-DWA algorithm. *Sci. Rep.* **2025**, *15*, 2099.
29. Liu, Z.; Li, J.; Wang, H. Integrating Improved A * and DWA Algorithms for Indoor Robot Path Planning. *Manuf. Autom.* **2025**, *47*, 51–58.
30. Wang, H.; Ma, X.; Dai, W.; Jin, W. Research on obstacle avoidance for mobile robots using improved DWA algorithm. *Comput. Eng. Appl.* **2023**, *59*, 326–332.
31. Ziyang, L.; Jie, Z.; Hao, Y.; Jing, L. Integrating improved RRT and DWA algorithms for mobile robot path planning. *Mech. Des. Manuf.* **2023**, *11*, 224–229. <https://doi.org/10.19356/j.cnki.1001-3997.20230525.033>.

32. Zhen, Y.; Li, J.; Yang, L.; Wang, Q. A distributed multi mobile robot path planning method that integrates security A* with DWA. *Control Eng.* **2024**, *31*, 2284–2295. <https://doi.org/10.14107/j.cnki.kzgc.20220516>.
33. Wang, H.; Song, G.; Chao, S.; Tan, Q.; Ge, C. Robot dynamic path planning based on beetle algorithm and DWA algorithm. *Manuf. Technol. Mach. Tools* **2024**, *12*, 21–29. <https://doi.org/10.19287/j.mtmt.1005-2402.2024.12.002>.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.