

Article

Optimized PAB-RRT Algorithm for Autonomous Vehicle Path Planning in Complex Scenarios

Jinbo Wang ¹, Weihai Zhang ^{1,*}, Jinming Zhang ², Wei Liao ¹ and Tingwei Du ¹

¹ School of Automotive Engineering, Shandong Jiaotong University, Jinan 250307, China; wangjinbo@sdjtu.edu.cn (J.W.); 23104013@stu.sdjtu.edu.cn (W.L.)

² School of Mechanical and Electrical Engineering, Weifang University of Science and Technology, Weifang 262700, China; zhangjinming@wfust.edu.cn

* Correspondence: 23104030@stu.sdjtu.edu.cn

Abstract

Path planning is a pivotal technology for autonomous vehicles, directly influencing driving safety and comfort. Developing algorithms adaptable to diverse scenarios is critical for ensuring the safe operation of autonomous driving systems and advancing their engineering applications. The existing Rapidly exploring Random Tree (RRT) algorithm has limitations such as low efficiency and tortuous, lengthy paths. To address these issues, this study proposes the PAB-RRT algorithm, which integrates probabilistic goal bias, adaptive step size, and bidirectional exploration into RRT. Comparative simulations were conducted to evaluate PAB-RRT against traditional RRT, RRT*, and single-strategy improved variants (A-RRT, P-RRT, B-RRT). Results show that in static multi-obstacle scenarios, PAB-RRT completes planning with 30 iterations (6.99% of traditional RRT), 0.1255 s computation time (21.9% of traditional RRT), and a 130.83 m path length (7.2% shorter than traditional RRT). In dynamic obstacle scenarios, it requires 19 iterations (0.0434 s) at the initial stage and 37 iterations (0.0861 s) after obstacle movement, with path length stably around 130 m. Overall, PAB-RRT outperforms traditional algorithms in exploration efficiency, path performance, and robustness in complex settings, better meeting the efficiency and reliability requirements of autonomous vehicle path planning under complex scenarios and providing a feasible reference for related technology.

Keywords: trajectory planning; RRT algorithm; biased sampling; expansion step size; bidirectional RRT

Academic Editor: Felipe Jiménez

Received: 1 January 2026

Revised: 30 January 2026

Accepted: 30 January 2026

Published: 2 February 2026

Copyright: © 2026 by the authors. 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 widespread adoption of artificial intelligence tools, autonomous vehicle technology has undergone continuous iteration and advancement [1], attracting growing attention from researchers [2,3]. Autonomous vehicles are currently deployed across diverse scenarios [4]. As a core decision-making module of autonomous driving systems, path planning directly determines driving safety, efficiency, and comfort, acting as a key enabler for the transformation from technical validation to large-scale commercialization of autonomous driving. Its primary aim is to rapidly create an optimal driving path that conforms to constraints in complex dynamic environments, integrating multi-source information such as real-time traffic conditions, road topology, and obstacle distribution.

Different application scenarios (e.g., urban congested roads, highways, closed campuses) impose differentiated demands for real-time response speed and obstacle avoidance robustness of trajectory planning, presenting multiple challenges including dynamic obstacle prediction, multi-objective conflict coordination, and adaptability to complex road conditions [5,6]. These practical demands and technical bottlenecks have driven researchers to explore more efficient and intelligent path planning algorithms and solutions [7].

Currently, research on autonomous driving path planning has developed a trend of multi-algorithm integration and multi-scenario adaptation. Trajectory planning algorithms like Dijkstra's algorithm and the A* algorithm are extensively utilized in static structured environments due to their advantages of simple principles and stable solution performance. However, their real-time performance and flexibility are difficult to meet the requirements in facing dynamically changing, complex traffic scenarios [8]. With the rise of intelligent algorithms, deep learning-based path planning models and multi-objective optimization algorithms have become research hotspots [9]. These methods can better handle environmental uncertainties and achieve dynamic path adjustment and global optimization, but some algorithms suffer from problems including elevated computational complexity and insufficient generalization ability in real-world operating scenarios [10,11]. In addition, path planning schemes integrating high-precision maps and real-time sensor perception data [12] can improve path accuracy, but they face practical challenges, including map update delays and multi-source data fusion conflicts, and related technologies still need further breakthroughs and improvements. In the application and exploration of the above-mentioned various algorithms, academia and industry have always strived to balance the three core objectives of planning efficiency, environmental adaptability, and path optimality, but have not yet found an ideal solution that can simultaneously take into account the real-time responsiveness of complex time-varying environmental scenarios and the quality of globally optimal paths. The structural dependence of traditional graph search algorithms, the high computing power threshold of intelligent learning-based algorithms, and the data synchronization problems of high-precision map fusion schemes together constitute the bottlenecks for the practical implementation of autonomous driving trajectory planning technology. For the typical multi-vehicle interaction scenario of unsignalized intersections, Viadero-Monasterio et al. [13] proposed a collaborative framework that integrates motion planning and robust output feedback trajectory tracking. Their research confirmed that generating collision-free paths and velocity curves solely through centralized planning algorithms is difficult to completely avoid intersection conflicts. It is necessary to design a tracking controller with linear matrix inequality (LMI) constraints to deeply match the feasibility and tracking accuracy of the planned trajectory, ultimately achieving a significant reduction in lateral deviation and velocity tracking error by 50%. This conclusion highlights the core logic of providing "low-error sensitivity" inputs for tracking control in the planning stage. Under the general engineering constraint of network-induced delay, Viadero-Monasterio et al. [14] further revealed the strong correlation between path characteristics and tracking stability – when the planned path curvature mutation frequency exceeds 5 Hz, even with event-triggered multiple input–multiple output (MIMO) linear parameter variation (LPV) control strategy, the vehicle roll angle overshoot will still increase by 40%. This study effectively alleviated the negative impact of delay by adapting the signal update frequency of tracking control to the dynamic characteristics of the planned trajectory, and clarified the design principle of "planning needs to reduce the sensitivity of tracking to the network environment". These studies all indicate that path planning algorithms not only need to solve the problem of "whether a path can be found" but also need to address the question of "whether the path can be stably tracked" [15]. In addition, the energy management strategy for fuel cell vehicles and the multi-condition vehicle speed prediction method

proposed by Lu et al. [16] provide important technical references for energy consumption optimization and complex scenario adaptability improvement in intelligent fuel cell vehicle path planning. Among the family of sampling-driven path planning algorithms, the Rapidly exploring Random Tree (RRT) algorithm has gradually become a key research direction to break through the above bottlenecks, thanks to its probabilistic completeness, adaptability to high-dimensional spaces, and ease of implementation. Its flexible, tree-like expansion structure is naturally suitable for path search in complex, unstructured environments.

The RRT algorithm has attracted significant attention in the field of intelligent vehicles [17] and is also widely applied to path planning tasks [18]. RRT improved through multiple methods can achieve effective path planning decisions by performing hierarchical improvements on the traditional RRT [19,20]. This approach enables rapid resolution of complex path planning problems. Additionally, the improved RRT acquires optimal strategies through interaction with complex environments, allowing it to maintain a high level of optimization capability even under dynamically changing driving conditions. For instance, Yao et al. [21] presented a modified goal-biased RRT optimization algorithm, which has certain limitations in application scenarios—it can only be used in static obstacle environments, i.e., scenarios where all environmental factors have deterministic probabilities. When compared with the traditional RRT algorithm, the RRT–Star algorithm [22,23] reselects and connects neighboring nodes with the lowest cost when adding new nodes, thereby lowering the total cost commencing from the initial point to the new point. By virtue of this unique mechanism, the RRT–Star algorithm can gradually generate paths close to the optimal standard [24]. However, when conducting node expansion in highly intricate environments, the algorithm exhibits a distinct drawback—a lack of clear goal orientation, which directly leads to sluggish convergence and inferior path quality of the final generated path [25]. Huang et al. [26] integrated the RRT* method with the Artificial Potential Field (APF) technique, introducing a goal target-preferential strategy and gravitational constituent elements for guidance to make the random tree grow toward the goal point. This improvement mitigates unproductive sampling of the computational scheme and enhances its efficiency. Nevertheless, the APF method [27,28] suffers from issues of local optimality, and the expansion of the random tree expansion—especially in complex unstructured environments—could result in the method being stuck in local optimal trapping problems, reducing path planning quality. Though these RRT improved variants [29,30] have optimized sampling mechanisms in various ways, their improvements are mostly concentrated on a single dimension, failing to fully explore and utilize environmental information. Therefore, such algorithms still exhibit certain constraints in addressing intricate environmental issues and achieving all-round path optimization.

In summary, to mitigate the defects of the traditional RRT method in autonomous driving trajectory planning—such as slow convergence, path redundancy, and poor adaptability to dynamic environments—existing improved algorithms have achieved targeted breakthroughs in their respective research dimensions: for convergence speed, goal-biased RRT algorithms reduce the proportion of invalid expansion through directed sampling; for path quality, the RRT* algorithm realizes iterative reduction of path costs through a neighborhood optimization mechanism; for adaptability to complex environments, RRT variants integrated with the APF method alleviate the blindness of random expansion through gravitational guidance. However, the improvement strategies of these algorithms are mostly single-point optimizations, rendering it arduous to simultaneously balance real-time responsiveness, safety, and trajectory optimality in multi-obstacle scenarios. To tackle the problems outlined above, this research puts forward the PAB-RRT algorithm, a multi-strategy fusion approach integrating probability target bias sampling,

adaptive step size, and bidirectional search. Probability target bias sampling helps the bidirectional RRT identify the correct search direction by focusing sampling points on the target area, avoiding misalignment of the two random trees in invalid regions during bidirectional search, and making the initial path form closer to the optimal direction. The adaptive step size balances safety and efficiency for the bidirectional RRT, using large step sizes in open areas to accelerate the approach of the two trees and small step sizes near obstacles for precise obstacle avoidance. The bidirectional RRT itself reduces the scope of single-tree unidirectional exploration through parallel bidirectional search, further improving planning efficiency. The three strategies work synergistically: the combination of probability target bias sampling and bidirectional RRT ensures planning real-time performance, while the collaboration of adaptive step size with the other two strategies balances obstacle avoidance safety and path smoothness, ultimately outputting a path that meets the “efficiency, performance, and quality” requirements of autonomous driving.

In conclusion, the main contributions of this study are summarized as follows:

1. An improved PAB-RRT algorithm is proposed, integrating three core optimization strategies—probability target bias sampling, adaptive expansion step size, and bidirectional RRT—to achieve multi-dimensional performance improvement.
2. A collision detection method integrating vehicle collision circles and offset lines is designed, and combined with the path smoothing strategy of node clipping and cubic B-splines to ensure that the planned path meets the dynamic constraints of vehicle driving. Meanwhile, two typical test scenarios (static multi-obstacle and dynamic obstacle scenarios) are constructed, forming a complete research framework from algorithm design to scenario verification.

The following is the structure of the subsequent content in this paper: Section 2 elaborates on the method for improving the RRT algorithm, elaborating on the design principles and implementation details of the three core optimization strategies to provide methodological support for the construction of the PAB-RRT algorithm. Section 3 introduces the traditional RRT and RRT* algorithms, clarifying the technical foundation and comparative benchmarks of the improved algorithm. Section 4 proposes path processing techniques, including key technologies such as vehicle driving environment modeling, vehicle collision detection, and path smoothing, to ensure that the planned trajectory satisfies the dynamic constraints of intelligent vehicle driving and practical application requirements. Section 5 conducts experimental research, designing comparative simulation experiments by constructing two typical scenarios: static multi-obstacle and dynamic obstacle environments. Section 6 summarizes the main conclusions.

2. Multi-Strategy Improved RRT Algorithm

2.1. Probability Target Bias Sampling

The conventional RRT algorithm utilizes uniform random sampling strategy [31], where the sampling point q_{rand} follows a uniform distribution $P_{\text{uni}}(q)$ in the state space $\mathcal{X}$. The formula is expressed as:

$$P_{\text{uni}}(q) = \begin{cases} \frac{1}{|\mathcal{X}_{\text{free}}|}, & q \in \mathcal{X}_{\text{free}} \\ 0, & q \in \mathcal{X}_{\text{obs}} \end{cases} \quad (1)$$

where $|\mathcal{X}_{\text{free}}|$ denotes the number of grids in the feasible region. However, uniform sampling lacks orientation towards the target point q_{goal} , resulting in low efficiency of random tree expansion towards the target. Especially in complex environments, a large quantity of sampling nodes are wasted in regions far from the target. To address this issue,

probability target bias sampling is introduced to make sampling points biased towards the target with a specific probability, thus enhancing path planning efficiency.

To this end, a hybrid sampling probability distribution $P_{min}(q)$ is designed, integrating uniform sampling and target-biased sampling:

$$P_{min}(q) = \alpha \cdot P_{uni}(q) + (1 - \alpha) \cdot P_{goal}(q) \quad (2)$$

In the formula, $\alpha \in [0, 1]$ is a weight coefficient that controls the proportion of the two sampling methods. A larger α indicates a higher proportion of uniform sampling and stronger exploration capability; a smaller α means stronger target bias and faster convergence. $P_{goal}(q)$ is the target-biased sampling probability, designed as a Gaussian distribution centered at the target point q_{goal} , with the formula:

$$P_{goal}(q) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{\|q - q_{goal}\|^2}{2\sigma^2}\right) \quad (3)$$

where σ represents the standard deviation of the Gaussian distribution, controlling the bias range; $\|q - q_{goal}\|$ denotes the Euclidean distance between the sampling node and the target point.

Since the grid map is discrete, the Gaussian distribution is discretized. For each feasible grid (i, j) , the distance to the target grid (i_g, j_g) is calculated as $d_{(i,j)} = \sqrt{(i - i_g)^2 + (j - j_g)^2}$. The sampling probability of this grid is:

$$P_{grid}(i, j) = \frac{\exp\left(-\frac{d_{(i,j)}^2}{2\sigma^2}\right)}{\sum_{(m,n) \in \mathcal{X}_{free}} \exp\left(-\frac{d_{(m,n)}^2}{2\sigma^2}\right)} \quad (4)$$

This method offers significant advantages in vehicle path planning. On the one hand, by guiding sampling points to approach the target, it effectively reduces the random tree's expansion in invalid regions, accelerates the random tree's growth towards the area of the target, and thereby greatly improves the convergence speed of the algorithm. On the other side, as more sampling nodes concentrate in the target direction, the finally generated trajectory is more approaching the ideal trajectory, significantly enhancing the quality of path planning. In practical applications, relevant parameters can be dynamically adjusted in light of varying environmental complexity and planning requirements, enabling the algorithm to maintain high efficiency across different scenarios.

2.2. Adaptive Expansion Step Size

The traditional RRT algorithm employs a fixed expansion step size [32], which often causes a trade-off between path planning effectiveness and safety. A too-large step size may cause collision detection failure, while a too-small step size significantly increases the number of sampling points and computational time. To resolve this conflict, an environment-aware adaptive expansion step size mechanism is proposed, which dynamically regulates the step size by analyzing the relative relationship between sampling points and obstacles in real-time, balancing path safety and planning efficiency.

Let $Step_{max}$ be the preset maximum step size of the trajectory planning algorithm, $Step_{min}$ the minimum safe step size, q_{rand} the current random sampling point, $q_{nearest}$ the nearest neighbor node, and $d(q_{nearest}, q_{rand})$ the Euclidean distance of the two nodes. The base expansion step size $Step_{base}$ is defined as:

$$Step_{base} = \min(d(q_{nearest}, q_{rand}), Step_{max}) \quad (5)$$

Let d_{obs} be the distance to the closest obstacle in the current expansion direction to $q_{nearest}$. An obstacle influence factor λ is adopted to dynamically regulate the step size:

$$\lambda = \begin{cases} \lambda_{min} & , d_{obs} \leq Step_{min} \\ \frac{d_{obs}}{Step_{max}} & , Step_{min} < d_{obs} < Step_{max} \\ \lambda_{max} & , d_{obs} \geq Step_{max} \end{cases} \quad (6)$$

When the obstacle is too close ($d_{obs} \leq Step_{min}$), λ takes the minimum value λ_{min} to force step size reduction. When the environment is open ($d_{obs} \geq Step_{max}$), λ takes the maximum value λ_{max} to maintain maximum step size efficiency.

Finally, integrating the base step size and the obstacle influence factor, the formula for the final expansion step size $Step_{final}$ is:

$$Step_{final} = \max(Step_{base} \times \lambda, Step_{min}) \quad (7)$$

2.3. Bidirectional RRT Extension

The traditional RRT algorithm adopts a unidirectional search strategy [33], constructing a random tree starting from the initial point and gradually expanding towards the target point. This approach exhibits low search efficiency in complex environments or high-dimensional spaces, especially when the initial point and target point are far apart—requiring extensive expansion of the random tree to find a feasible path. The bidirectional RRT extension method constructs random trees simultaneously from the initial trajectory point and the target trajectory point, enabling the two random trees to expand towards one another until they meet, thereby finding a path establishing a connection between the initial and the target node.

First, initialize two random trees T_a and T_b with q_{start} and q_{goal} as root nodes, respectively. Then, perform heuristic sampling to generate a sampling point q_{rand} , where q_{goal} is selected to serve as the sampling point with a specific probability to improve the algorithm's convergence speed. The probability calculation formula is:

$$P = \alpha \times e^{-\beta \times d} \quad (8)$$

where α and β are constants, and d represents the distance from the closest node in the current random tree to the target path point. Next, expand the nodes of T_a and T_b respectively. The expansion method is analogous to that of the traditional RRT algorithm, but with the addition of searching for the other tree during expansion. Subsequently, perform inter-tree connection detection: when the distance between the expanded point q_{new_a} in T_a and the point q_{near_b} in T_b is smaller than a set threshold, determine if they can be connected. If yes, a feasible path is found. Figure 1 shows a principle schematic diagram of the above improved method. The left side of the figure shows the RRT sampling process, while the right side corresponds to solutions for its drawbacks, such as too many invalid nodes, redundant search, and long one-way RRT time.

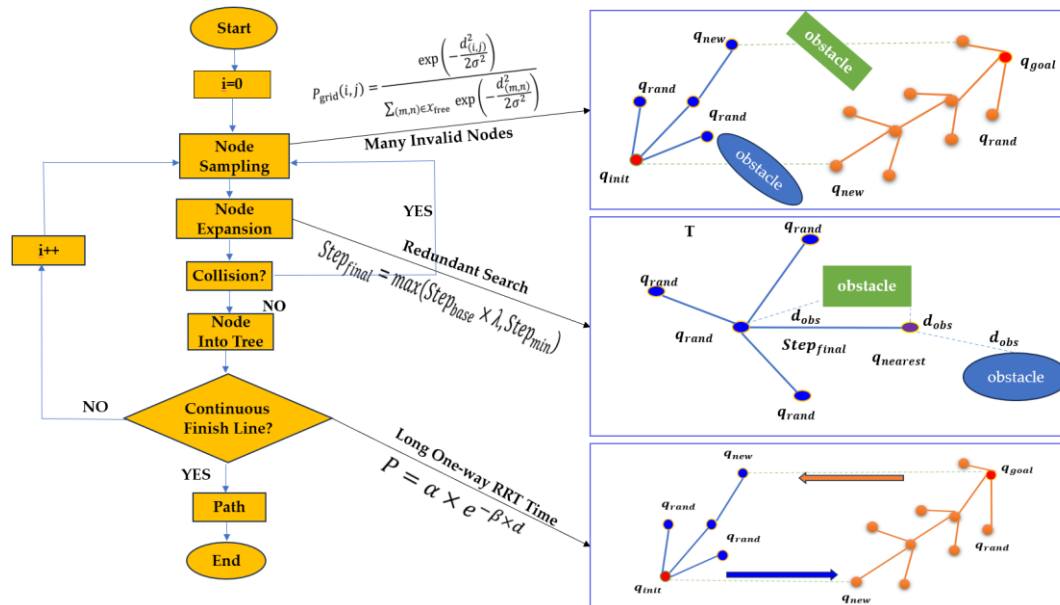


Figure 1. Schematic of the PAB-RRT algorithm.

3. Principles of Traditional RRT and RRT* Algorithms

3.1. RRT Algorithm

In this study, the state domain of the environment where the vehicle operates is assumed to be X , with $N \in R^N$, where N represents the state space dimension, which is typically 2 or 3. In this study, N is set to 2. The obstacle region in the environment is denoted as X_{ob} , and the obstacle-free region is X_f , satisfying the following relationships:

$$\begin{aligned} X_f \cup X_{ob} &= X \\ X_f \cap X_{ob} &= \emptyset \end{aligned} \quad (9)$$

The root node q_{init} is used to represent the vehicle's starting position, and q_{goal} denotes the target endpoint. According to constraints, both the start and end points must be located in the obstacle-free region, $q_{init} \in X_f$ and $q_{goal} \in X_f$. Define the generated set of random trees as T , with the start point q_{init} acting as the root node of the search tree structure. The branches of the random tree T are expanded according to a preset step size s . In the course of expansion, a node is randomly selected as the sampling point q_{rand} in the region free of obstacles X_f of the state space. Subsequently, the distance between each node within the random tree T and the sampling point q_{rand} is calculated using the distance function $L(q_1, q_2)$, and the node with the minimum distance to q_{rand} is selected and denoted as $q_{nearest}$. Here, the Euclidean distance calculation function is assumed as the evaluation function between nodes, and the expression for the path length between two connected nodes is defined as:

$$L(q_1, q_2) = \sqrt{(x_{q1} - x_{q2})^2 + (y_{q1} - y_{q2})^2} \quad (10)$$

After determining $q_{nearest}$, the closest node to the sampling node, the random tree T needs to be further expanded towards the sampling point q_{rand} . Specifically, $q_{nearest}$ is taken to serve as the starting position of extension, and a novel node q_{new} is obtained by stretching a step size s in the direction from $q_{nearest}$ to q_{rand} . Since the feasibility of the position of q_{new} in the state space is not yet clear, collision detection must be performed on the line segment between $q_{nearest}$ and q_{new} : if the detection result shows no collision, q_{new} is incorporated into the random tree T as a novel node; if a collision is identified, the point is rejected and sampling and expansion are repeated. The above process is iterated

until the random tree T includes the target point q_{goal} or the preset maximum iterative times are reached, at which point the sampling operation stops. Its principle is depicted in Figure 2.

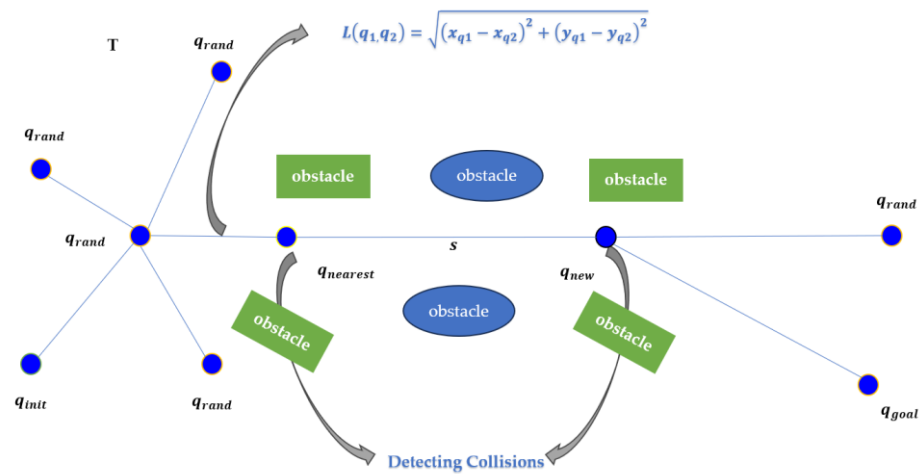


Figure 2. Schematic drawing of the RRT algorithm.

3.2. RRT* Algorithm

To overcome the defects of the traditional RRT algorithm, such as non-optimal paths and limited convergence efficiency [34,35], the Rapidly exploring Random Tree Star (RRT*) algorithm was thus born. This algorithm introduces path reconnection and neighborhood node optimization mechanisms, gradually approaching the globally optimal solution for the generated path through local search around nodes and path cost updates. It has become a classic vehicle path planning algorithm that balances probabilistic completeness and asymptotic optimality.

As shown in Figure 3, the core idea of the RRT* algorithm is to gradually optimize the path cost through the collaborative mechanism of local optimization and global search during the random tree expansion process. Its basic flow adds two key steps—neighborhood node search and path reconnection—on the basis of the traditional RRT [36,37]:

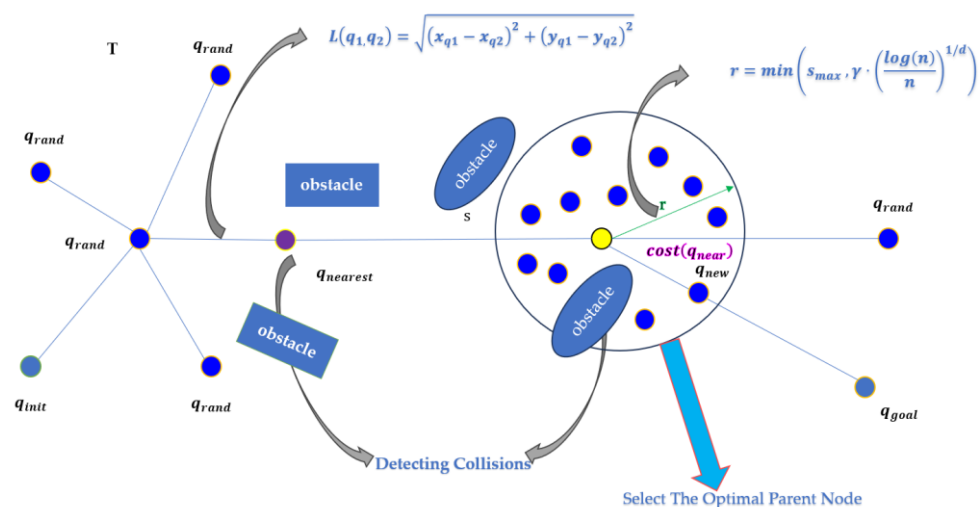


Figure 3. RRT* principle diagram.

(1) Neighborhood Node Search

Define a neighborhood range centered at q_{new} with radius r (r is usually dynamically adjusted according to environmental complexity. For instance, $r = \min(s_{max}, \gamma \cdot (\frac{\log(n)}{n})^{1/d})$, where n represents the node count within the tree, d is the spatial dimension, and γ denotes a constant. All nodes in the random tree T located within this neighborhood are selected to form the neighborhood node set N_{near} . The core purpose of this step is to find potential optimal parent nodes around q_{new} , nodes that result in a lower path cost when connecting to q_{new} through existing nodes in the neighborhood.

(2) Path Reconnection

After selecting the parent node of q_{new} , traverse the neighborhood set N_{near} again: for each node q_{near} , calculate the cost $cost(q_{near})$. In the absence of a collision between q_{new} and q_{near} , update the parental node of q_{near} to q_{new} , and synchronously update the path costs of q_{near} and all its child nodes.

4. Vehicle and Path Processing

4.1. Vehicle Driving Environment Modeling

Within the domain of vehicle route planning, grid maps are widely used as a classic environmental modeling method due to their intuitiveness and efficiency [38]. A grid map discretizes the continuous space of vehicle operation into regularly arranged grid cells. Each grid represents a region in the actual environment, and different attribute values are assigned to describe the characteristics of the region, such as obstacle distribution and terrain information. This discretization approach converts complex real-world environments into digital models that are easy for computers to process, providing a structured data foundation for subsequent path planning methods.

As shown in Figure 4, the grid map discretizes the vehicle driving area into regular grids, transforming the continuous physical space into uniform-sized grid cells. Grid maps offer distinct advantages. Firstly, they can intuitively present environmental information, facilitating understanding and analysis by researchers. Secondly, the grid-based data structure is highly compatible with computer storage and processing methods, enabling path planning algorithms to quickly retrieve and process map information, thereby improving algorithm execution efficiency. Thirdly, grid maps exhibit good scalability, allowing for convenient integration of multi-source information to adapt to complex and dynamic environments.

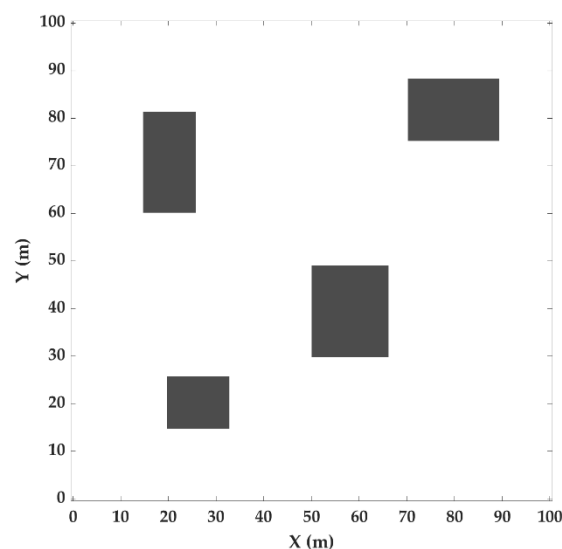


Figure 4. Grid-based map.

The bottom-left corner in the figure serves as the start point q_{start} for vehicle path planning, which is the initial position of the path search. The top-right corner acts as the target endpoint q_{goal} . To describe the environmental model of the grid map mathematically, assume the map contains $M \times N$ grids. The environmental state space $\mathcal{X}$ may be expressed as:

$$\mathcal{X} = \{ (i, j) \mid i = 1, 2, \dots, M; j = 1, 2, \dots, N; s(i, j) = 0 \text{ or } 1 \} \quad (11)$$

In the formula, (i, j) denotes the grid's row and column indices. $s(i, j) = 0$ indicates that the grid is white and passable; otherwise, $s(i, j) = 1$ means the grid is impassable. This provides environmental constraints for the operations of sampling and expansion of the subsequent RRT algorithm.

4.2. Vehicle Processing Strategy

To address the problem with obstacle collisions arising from the failure to consider the actual vehicle size when expanding nodes, a hybrid collision detection method combining vehicle collision circles and offset line collision is adopted. The principle of collision detection is demonstrated in Figure 5a,b, where the expanded circle in Figure 5b is an abridged version of Figure 5a.

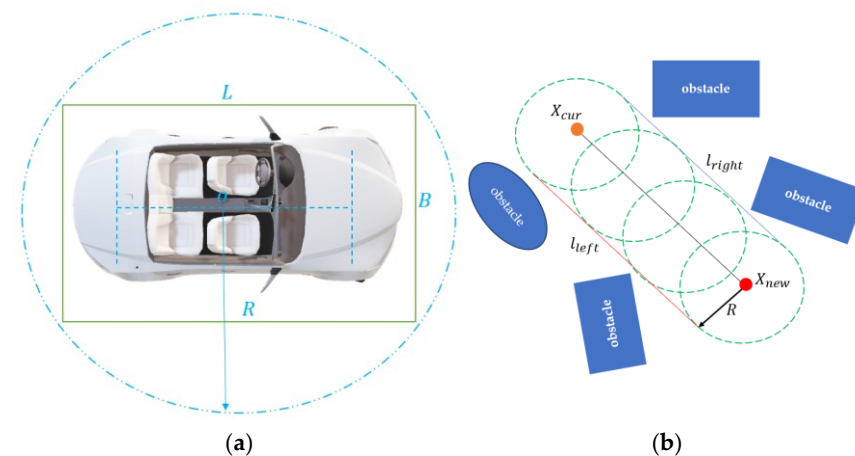


Figure 5. Vehicle processing strategy. (a) Vehicle collision processing, (b) offset line collision detection.

As shown in Figure 5, the calculation of the vehicle expanded circle is given by Equation (12):

$$R = (1 + \delta) \times \frac{\sqrt{L^2 + B^2}}{2} \quad (12)$$

where δ is the relaxation factor, L is the vehicle length, and B is the vehicle width. To ensure that the connection line between newly generated nodes does not collide with obstacles, a left–right offset line method is used to logically judge obstacles between two points, as shown in Equation (13):

$$flag = \begin{cases} 1, & \text{if } (l_{left} == 1, l_{right} == 1) \\ 0, & \text{else} \end{cases} \quad (13)$$

In the formula, l_{left} is the left offset line, l_{right} is the right offset line, and flag is the collision flag.

As shown in Figure 5b, l_{left} is the red-colored line, l_{right} is the blue line, and the black line is the extended tree connection obtained without colliding with obstacles. R is the vehicle expanded size, and X_{cur} is the current node. When neither offset line between X_{cur} and X_{new} collides against obstacles, the flag is set to 1, indicating passability.

4.3. Path Smoothing Processing

Owing to the random sampling characteristic of the RRT algorithm in the path search process, directly producing a path from the target point back to the start point may result in an excessively tortuous trajectory. Such a path may fail to satisfy the maximum curvature requirement in vehicle steering, generate redundant path points [39], and thereby increase path costs. Referring to the node clipping method in Figure 6, beginning from the target point, an attempt is made to connect to the start point while checking for intersections between this line segment and obstacles. If there is no intersection, all path points between these two nodes can be eliminated, and the two nodes are directly connected by a line segment as part of the optimal trajectory. When an intersection is present, an attempt is made to connect node q_2 to the target node, and the intersection check is conducted repeatedly. During the course of this process, it is also essential to consider whether the connection line between nodes meets the maximum steering angle constraint of the vehicle. Only when the line segment has no intersection under the obstacle environment and complies with the steering angle constraint is the node regarded as the next path point. This process is iterated until an optimal path that neither intersects with obstacles nor violates the vehicle steering constraint is found. This method ensures that, in the case of using B-spline curves for path smoothing, the curvature of the path does not exceed the vehicle's maximum curvature limit. After such optimization, the final path, as shown by the solid line in Figure 6, is both concise and meets the vehicle's driving requirements.

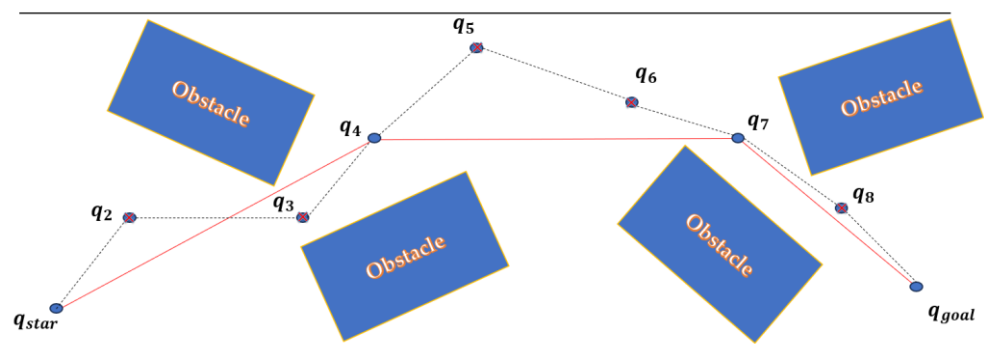


Figure 6. Node clipping method.

Following the aforementioned path clipping, the path still suffers from the issue of discontinuous curvature. Consequently, cubic B-spline curves are used for path smoothing and achieving continuous curvature of the planned trajectory. When the count of control points is $n + 1$, a K -th order B-spline curve is defined.

Here, $P(t)$ are the control points, and $B_i, K(t)$ are the cubic B-spline's basis functions obtained recursively using the DeBoor–Cox formula [40]:

$$P(t) = [p_0, p_1, \dots, p_n] \begin{bmatrix} B_{0,K}(t) \\ B_{1,K}(t) \\ \dots \\ B_{n,K}(t) \end{bmatrix} = \sum_{i=0}^n p_i B_{i,K}(t) \quad (14)$$

$$\begin{cases} B_{i,0}(t) \begin{cases} 1, & t_i \leq t \leq t_{i+1}, \\ 0, & \text{Otherwise,} \end{cases} & K = 1 \\ B_{i,3}(t) = \frac{(t - t_i)B_{i,2}(t)}{t_{i+3} - t_i} + \frac{(t_{i+4} - t)B_{i+1,2}(t)}{t_{i+4} - t_{i+1}}, & K \geq 2 \end{cases} \quad (15)$$

Three uniform B-splines are employed to smooth the planned path, with the node repetition degree at both ends set to 3. The basis functions can then be represented as:

$$\begin{cases} B_{0,3}(t) = \frac{1}{6}(1-t)^3 \\ B_{1,3}(t) = \frac{1}{6}(3t^3 - 6t^2 + 4) \\ B_{2,3}(t) = \frac{1}{6}(-3t^3 + 3t^2 + 3t + 1) \\ B_{3,3}(t) = \frac{1}{6}t^3 \end{cases} \quad (16)$$

In this context, the range of the parameter node vector is set to $[0, 1]$, and the expression for the third-order quasi-uniform B-spline curve can be derived as:

$$P(t) = P_0 B_{0,3}(t) + P_1 B_{1,3}(t) + P_2 B_{2,3}(t) + P_3 B_{3,3}(t), t \in [0, 1]. \quad (17)$$

4.4. Jump Redundancy Removal

The jump redundancy removal algorithm realizes redundant point elimination through geometric distance judgment. Let the original path point set be $P = \{p_0, p_1, \dots, p_n\}$, and initialize the optimized path set as $Q = \{p_0\}$. Starting from the start point p_0 , sequentially detect the distance along a straight line between each subsequent point p_i and the current terminal point of the path q_{last} . If $d(q_{last}, p_i) \leq \epsilon$ (where ϵ is the distance threshold), p_i is judged as a redundant point and skipped; otherwise, p_i is added to the optimized path set Q , and $q_{last} = p_i$ is updated.

After inputting the initial path, each node is judged based on distance to output the optimized path. The application of jump redundancy removal technology in vehicle path planning eliminates redundant points, significantly decreases the number of path nodes, and makes the path smoother. This helps the vehicle execute path planning instructions more efficiently and improves the vehicle's response speed in dynamic environments.

4.5. Interpolation Point Compensation

Although the integrated PAB-RRT algorithm optimizes the distribution of path nodes to a certain extent, affected by the search strategy and environmental modeling accuracy, the planned path may have problems such as sparse nodes and missing key position information. This can lead to insufficient path details and potential driving deviations during vehicle operation due to incomplete path information. Interpolation point compensation technology supplements missing path information by inserting new nodes between path nodes, making the path more accurate and complete. A flow diagram of path processing is presented in Figure 7.

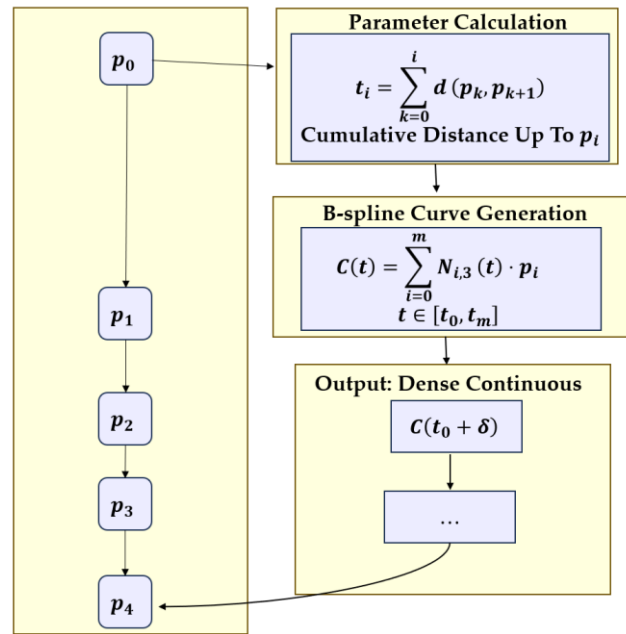


Figure 7. Path processing flow chart.

To solve the problem of discontinuous trajectories caused by sparse path points, let the path points after redundancy removal be $Q = \{p_0, p_1, \dots, p_m\}$. Define the parameter vector $t = [t_0, t_1, \dots, t_m]$, where $t_i = \sum_{k=0}^i d(p_k, p_{k+1})$; construct the cubic B-spline basis function $N_{i,3}(t)$, and the interpolation curve expression is:

$$C(t) = \sum_{i=0}^m N_{i,3}(t) \cdot p_i, \quad t \in [t_0, t_m] \quad (18)$$

Sample the parameter t with step size δ to generate the compensation point sequence $C(t_0 + \delta), C(t_0 + 2\delta), \dots, C(t_m)$.

5. Experimental Research

The experimental environment is a 2D static grid map with fixed rectangular obstacles. The start point is assigned to (5,5) and the end point to (90,90). The computer used in the experiment is equipped with an AMD Ryzen 7 7735H CPU, with a base frequency of 3201 MHz and 16GB of memory. All algorithms are run under the same start point, end point, and obstacle conditions to ensure the fairness of experimental comparisons. The main parameter settings are as follows: maximum number of iterations = 3000, step size = 2, and target region determination threshold = 3. A-RRT refers to the RRT algorithm with adaptive step size, B-RRT denotes the RRT algorithm with bidirectional search, and P-RRT represents the RRT algorithm with probability target bias sampling.

5.1. Multi-Obstacle Scenario

In order to validate the adaptability and stability of the integrated PAB-RRT algorithm in multi-obstacle environments, a multi-obstacle scenario is constructed for simulation testing. Based on a 2D static grid map, multiple rectangular obstacles are randomly distributed in the map to simulate complex obstacle avoidance scenarios encountered by autonomous vehicles on urban roads, such as continuous buildings, multiple groups of roadblocks, or temporary construction areas. This scenario is closer to actual driving conditions with dense obstacles. The experiment adheres to the principle of controlling variables from the simple obstacle scenario, ensuring the rigor of performance comparison between the traditional RRT algorithm, RRT*, A-RRT, B-RRT, P-RRT, and PAB-RRT

algorithms. Through this scenario, the path search capability, obstacle avoidance precision, and computational efficiency of the multi-strategy improved algorithm under the constraints of multi-obstacle interaction are analyzed, and the synergistic effect of the multi-strategy integrated optimized RRT in multi-obstacle environments is verified. Figure 8 illustrates the simulation results.

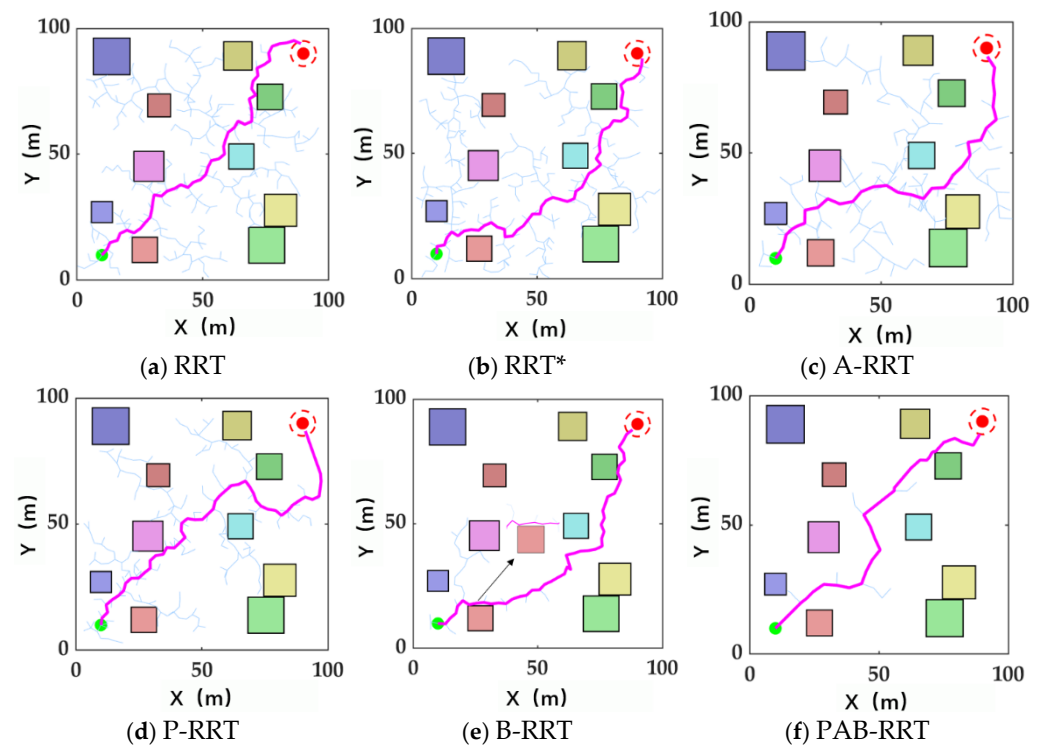


Figure 8. Simulation results of path planning in multi-obstacle scenarios.

In Figure 8, the geometric shapes represent obstacles, the green circular shape in the bottom-left corner is the start point, the red circular shape in the top-right corner is the end point, and the pink path is the trajectory from the start to the end point.

Figure 9 presents a detailed quantification of the simulation data, clearly revealing the performance differences between the traditional RRT algorithm, RRT*, P-RRT, A-RRT, B-RRT, and PAB-RRT in complex multi-obstacle environments. With respect to the number of iterations, the PAB-RRT algorithm completes path planning with only 30 iterations, which is much lower than the 429 iterations of the traditional RRT and 436 iterations of RRT*. Even the single-strategy improved algorithms—A-RRT (242 iterations), P-RRT (480 iterations), and B-RRT (73 iterations)—require far more iterations than PAB-RRT. This data intuitively reflects the significant improvement in algorithm search efficiency brought by multi-strategy integration: the orientation of probability target bias sampling and the parallel search of bidirectional RRT greatly reduce invalid iterations. With regard to path length, the path planned by PAB-RRT is 130.83 m, which is 7.2% shorter than the 141.00 m of the traditional RRT and 12.8% shorter than the 150.00 m of RRT*. It also outperforms A-RRT (143.93 m), P-RRT (159.00 m), and B-RRT (139.71 m). The precise adjustment of the adaptive expansion step size in obstacle areas and the smooth optimization of post-path processing enable PAB-RRT to effectively reduce the path length in complex obstacle environments. Regarding computational time, PAB-RRT takes only 0.1255 s, which is 21.9% of the traditional RRT (0.5702 s) and RRT* (0.5717 s), and significantly lower than A-RRT (0.3439 s), P-RRT (0.4856 s), and B-RRT (0.2213 s). The synergistic effect of multiple strategies greatly reduces the computational redundancy of the algorithm, enabling it to have stronger real-time performance while ensuring path quality, fully

meeting the efficiency requirements for path planning for autonomous vehicles in multi-obstacle scenarios.

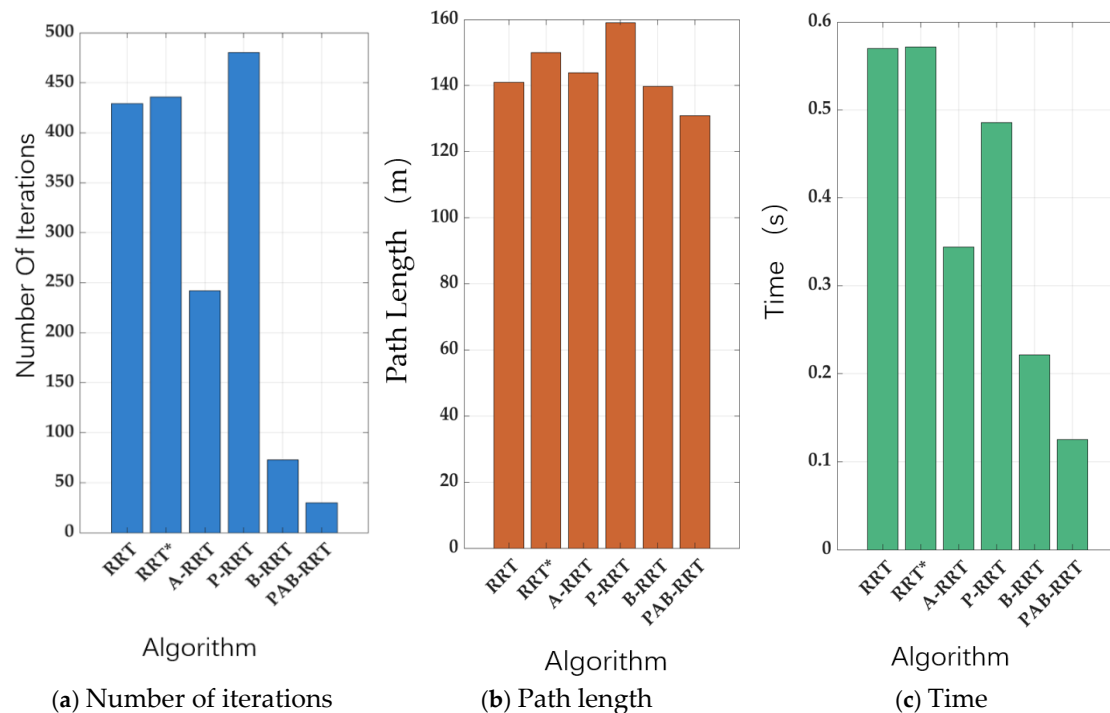
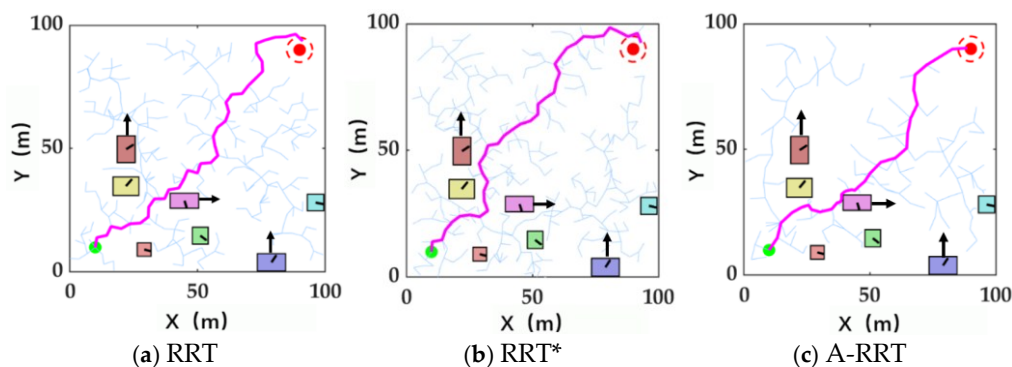


Figure 9. Data comparison of different algorithms.

5.2. Dynamic Obstacle Scenario

To further validate the practicality of the PAB-RRT algorithm in dynamic traffic scenarios, a dynamic obstacle scenario is constructed for simulation testing. Based on a 2D grid map, while maintaining the same basic conditions as the static scenario, multiple groups of dynamic obstacles moving along preset trajectories are introduced—such as other moving vehicles or objects temporarily crossing the road—to simulate the dynamic traffic flow environment encountered by autonomous vehicles on actual roads. By updating obstacle position information in real-time, the experiment tests the response speed, path adjustment capability, and obstacle avoidance safety of the traditional RRT algorithm, RRT*, and the multi-strategy improved RRT algorithm in dynamic environments. It focuses on analyzing the real-time capability and dependability of the improved algorithm in path planning under dynamic obstacle interaction. Figure 10 illustrates the simulation results.



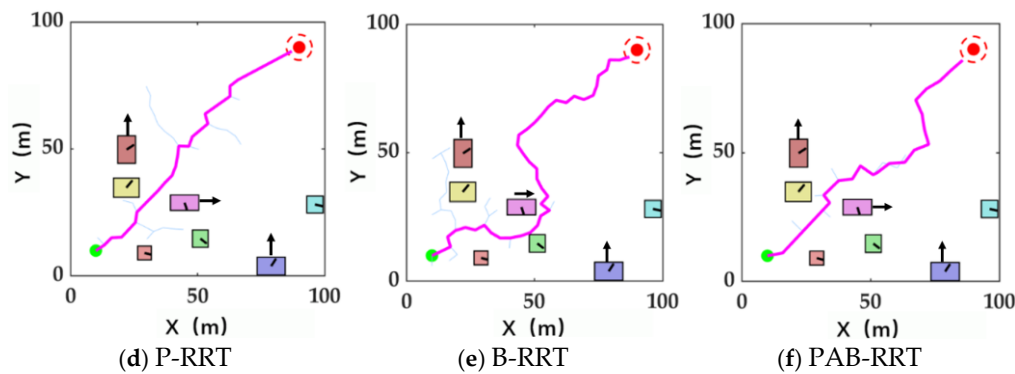


Figure 10. Path simulation diagrams under initial distribution of dynamic obstacles.

The experiment conducts dynamic scenario testing in two phases: the first phase is the initial distribution state of dynamic obstacles, where dynamic obstacles are at their initial positions (e.g., stationary in a specific lane of the road waiting to start, or at the initial moment of uniform movement). The algorithm needs to complete the first path planning under this initial distribution. Figure 10 presents the corresponding simulation results, where Figure 10a–f represent the planned paths of the traditional RRT, RRT*, A-RRT, P-RRT, B-RRT, and PAB-RRT algorithms in the initial state, respectively. The rectangles indicate the initial positions of dynamic obstacles, the green circles are the start points, the red circles are the end points, and the planned paths are denoted by the red line segments. The second phase is the state after the movement evolution of dynamic obstacles. After the initial planning is completed, dynamic obstacles move along preset trajectories (e.g., some obstacles move forward along the lane, some complete lane changes across lanes, and some adjust their driving speed) to simulate the dynamic changes in obstacle positions over time. After the obstacles move to new stable positions, the algorithm is triggered to re-plan the path. The corresponding simulation results are shown in Figure 12, where Figure 12a–f correspond to the algorithm types in Figure 11 one by one. The positions of the rectangles are significantly offset compared to Figure 10, intuitively reflecting the movement process of dynamic obstacles.

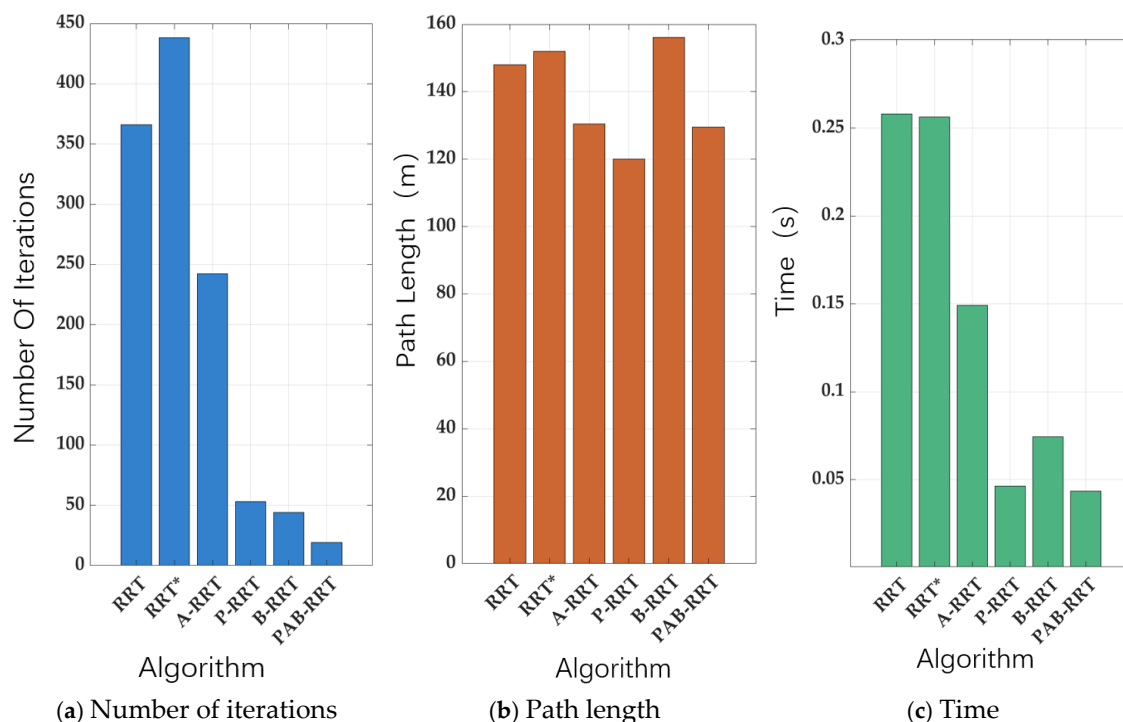


Figure 11. Path simulation comparison under initial distribution of dynamic obstacles.

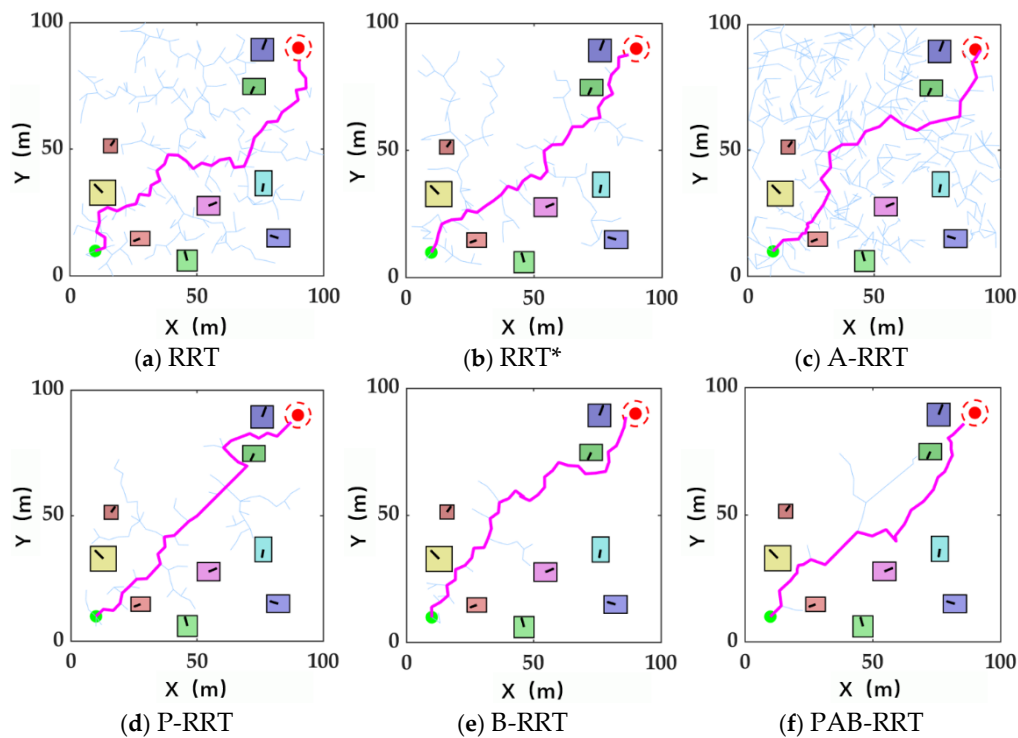


Figure 12. Path simulation diagrams after movement evolution of dynamic obstacles.

Based on the simulation result data in Figure 13, it can be known that the PAB-RRT algorithm exhibits notable advantages in dynamic scenarios. In the initial distribution phase of dynamic obstacles, it completes path planning with only 19 iterations, which is much lower than the 366 iterations of the traditional RRT and 438 iterations of RRT*. The corresponding computational time is only 0.0434 s, accounting for 16.8% of the traditional RRT (0.2580 s), and also outperforming the single-strategy improved A-RRT (0.1492 s) and B-RRT (0.0743 s), slightly higher than P-RRT (0.0463 s). At the same time, the path length planned in this phase is 129.50 m, which is 12.5% shorter than the 148.00 m of the traditional RRT and superior to the 152.00 m of RRT*. After the dynamic obstacles complete their movement evolution and enter a new distribution state, PAB-RRT still maintains high efficiency, completing path re-planning with only 37 iterations—far fewer than the 389 iterations of the traditional RRT and 540 iterations of A-RRT. The re-planning time is only 0.0861 s, which is much lower than the traditional RRT (0.3127 s), A-RRT (0.4452 s), and also less than RRT* (0.1810 s) and P-RRT (0.1368 s). Its re-planned path length is 129.71 m, which is 16.8% shorter than the 156.00 m of the traditional RRT, and superior to RRT* (132.00 m) and B-RRT (141.29 m). Overall, the algorithm achieves fast path planning and re-planning in dynamic scenarios while ensuring path economy, effectively adapting to the real-time obstacle avoidance and path update needs of autonomous vehicles in dynamic traffic flow environments.

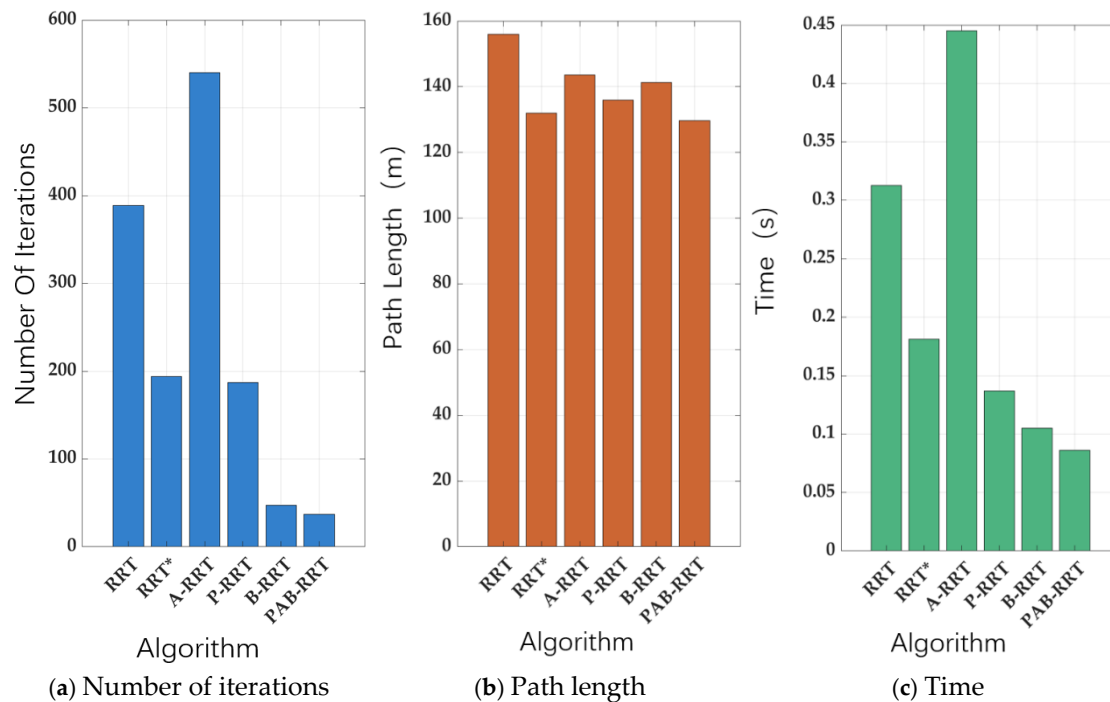


Figure 13. Path simulation comparison after movement evolution of dynamic obstacles.

6. Conclusions

To solve the defects of the traditional RRT algorithm in autonomous driving trajectory planning—such as low search efficiency, high path redundancy, and inadequate adaptability in dynamic environments—this paper puts forward the PAB-RRT algorithm, which integrates probability target bias sampling, adaptive expansion step size, and bidirectional RRT. The efficacy and predominance of the algorithm are verified through multi-scenario simulation tests.

Probability target bias sampling combines uniform distribution and Gaussian target-biased distribution, solving the problem of the traditional RRT's lack of goal orientation in uniform sampling and significantly reducing invalid sampling regions. The adaptive expansion step size dynamically adjusts the expansion step based on the distance to obstacles, achieving a balance between planning efficiency and obstacle avoidance safety. Bidirectional RRT shortens the search range of the random tree through parallel bidirectional search, further improving path convergence speed. Meanwhile, combined with the collision detection method integrating vehicle collision circles and offset lines, as well as the path smoothing strategy using node clipping and cubic B-splines, the PAB-RRT algorithm can generate smooth paths that meet the dynamic constraints of vehicle driving, effectively compensating for the excessive tortuosity of paths produced by the traditional RRT.

From the experimental results, in static multi-obstacle scenarios, the PAB-RRT algorithm completes path planning with only 30 iterations (far fewer than the 429 of traditional RRT and 436 of RRT*), a computational time of merely 0.1255 s (6.99% of traditional RRT), and a planned path length of 130.83 m (7.2% shorter than traditional RRT), demonstrating excellent search efficiency and path performance. In dynamic obstacle scenarios, PAB-RRT maintains leading performance both in the initial distribution phase and the post-movement re-planning phase—initial planning takes 19 iterations and 0.0434 s, while dynamic re-planning requires 37 iterations and 0.0861 s, with path length stably around 130 m—far superior to traditional algorithms. Thus, PAB-RRT can meet the real-time obstacle avoidance and path update needs of autonomous vehicles in dynamic traffic flow.

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

Funding: This research was funded by the Research and Industrialization of Key Technologies for Unmanned Intelligent Sanitation Vehicles Based on Data Fusion (grant number: 2025TSGCCZZB0653). Weifang City Science and Technology Development Plan Projects (College and University Section; grant number: 2025GX037).

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

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

Abbreviations

The following abbreviations are used in this manuscript:

RRT	Rapidly exploring random tree
PAB	Probabilistic goal bias, adaptive step size and bidirectional exploration

References

- Lu, D.; Hu, D.; Wang, J.; Wei, W.; Zhang, X. A data-driven vehicle speed prediction transfer learning method with improved adaptability across working conditions for intelligent fuel cell vehicle. *IEEE Trans. Intell. Transp. Syst.* **2025**, *26*, 10881–10891.
- Parekh, D.; Poddar, N.; Rajpurkar, A.; Chahal, M.; Kumar, N.; Joshi, G.P.; Cho, W. A review on autonomous vehicles. Progress, methods and challenges. *Electronics* **2022**, *11*, 2162.
- Betz, J.; Zheng, H.; Liniger, A.; Rosolia, U.; Karle, P.; Behl, M.; Mangharam, R. Autonomous vehicles on the edge: A survey on autonomous vehicle racing. *IEEE Open J. Intell. Transp. Syst.* **2022**, *3*, 458–488.
- Othman, K. Exploring the implications of autonomous vehicles: A comprehensive review. *Innov. Infrastruct. Solut.* **2022**, *7*, 165.
- Cheng, C.; Sha, Q.; He, B.; Li, G. Path planning and obstacle avoidance for AUV: A review. *Ocean. Eng.* **2021**, *235*, 109355.
- Yang, H.; He, Y.; Xu, Y.; Zhao, H. Collision avoidance for autonomous vehicles based on MPC with adaptive APF. *IEEE Trans. Intell. Veh.* **2023**, *9*, 1559–1570.
- Almusawi, A.; Albdairi, M.; Qadri, S.S.S.M. Integrating Autonomous Vehicles (AVs) into Urban Traffic: Simulating Driving and Signal Control. *Appl. Sci.* **2024**, *14*, 8851.
- Sriramulu, R.; Yadav, A.; Pal, S.B. Fast and efficient indoor navigation: A hybrid pathfinding approach using rapidly-exploring random tree (RRT)-connect and Dijkstra's algorithm. *PeerJ Comput. Sci.* **2025**, *11*, e3028.
- Bogyrbayeva, A.; Meraliyev, M.; Mustakhov, T.; Dauletbayev, B. Machine learning to solve vehicle routing problems: A survey. *IEEE Trans. Intell. Transp. Syst.* **2024**, *25*, 4754–4772.
- Fu, F.; Chen, Q.; Zhang, C.; Liu, J.; Han, Z. Deep learning path planning method by using proximity information fusion. *J. Nonlinear Convex Anal.* **2024**, *25*, 1469–1479.
- Tang, R.; Qi, L.; Ye, S.; Li, C.; Ni, T.; Guo, J.; Liu, H.; Li, Y.; Zuo, D.; Shi, J.; et al. Three-Dimensional Path Planning for AUVs Based on Interval Multi-Objective Secretary Bird Optimization Algorithm. *Symmetry* **2025**, *17*, 993.
- Liu, X.; Gong, G.; Hu, X.; Shang, G.; Zhu, H. Cognitive enhancement of robot path planning and environmental perception based on gmapping algorithm optimization. *Electronics* **2024**, *13*, 818.
- Viadero-Monasterio, F.; Meléndez-Useros, M.; Zhang, N.; Zhang, H.; Boada, B.L.; Boada, M.J.L. Motion planning and robust output-feedback trajectory tracking control for multiple intelligent and connected vehicles in unsignalized intersections. *IEEE Trans. Veh. Technol.* **2025**, *1*, 1–13.
- Viadero-Monasterio, F.; Nguyen, A.T.; Lauber, J.; Boada, M.J.L.; Boada, B.L. Event-triggered robust path tracking control considering roll stability under network-induced delays for autonomous vehicles. *IEEE Trans. Intell. Transp. Syst.* **2023**, *24*, 14743–14756.
- Zhang, W.; Wang, J.; Pang, T. Research on the Performance of Vehicle Lateral Control Algorithm Based on Vehicle Speed Variation. *World Electr. Veh. J.* **2025**, *16*, 259.

16. Lu, D.; Yi, F.; Hu, D.; Li, J.; Yang, Q.; Wang, J. Online optimization of energy management strategy for FCV control parameters considering dual power source lifespan decay synergy. *Appl. Energy* **2023**, *348*, 121516.
17. Wu, Z.; Meng, Z.; Zhao, W.; Wu, Z. Fast-RRT: A RRT-based optimal path finding method. *Appl. Sci.* **2021**, *11*, 11777.
18. Wang, H.; Lai, H.; Du, H.; Gao, G. IBPF-RRT*: An improved path planning algorithm with Ultra-low number of iterations and stabilized optimal path quality. *J. King Saud Univ.-Comput. Inf. Sci.* **2024**, *36*, 102146.
19. Ganesan, S.; Ramalingam, B.; Mohan, R.E. A hybrid sampling-based RRT* path planning algorithm for autonomous mobile robot navigation. *Expert Syst. Appl.* **2024**, *258*, 125206.
20. Liao, B.; Wan, F.; Hua, Y.; Ma, R.; Zhu, S.; Qing, X. F-RRT*: An improved path planning algorithm with improved initial solution and convergence rate. *Expert Syst. Appl.* **2021**, *184*, 115457.
21. Yao, X.Y.; Wang, X.; Wang, C. Research on 3D Path Planning of UAVs Based on Improved Goal-bias RRT. *Flight Dyn.* **2025**, *43*, 69–75.
22. Guan, T.; Han, Y.; Kong, M.; Wang, S.; Feng, D.; Yang, W. An improved artificial potential field with RRT star algorithm for autonomous vehicle path planning. *Sci. Rep.* **2025**, *15*, 16982.
23. Xue, W.; Wang, B.; Huang, X.; Yang, B.; Wen, Z.; Zhang, H.; Li, S. Spacecraft attitude maneuver planning with multi-sensor pointing constraints using improved RRT-star algorithm. *Adv. Space Res.* **2023**, *72*, 1485–1495.
24. Senthilnathan, R. Quicker Path planning of a collaborative dual-arm robot using Modified BP-RRT* algorithm. *Int. J. Comput. Commun. Control* **2024**, *19*.
25. Liu, T.; Chen, X.; He, M.; Fu, X.; Wu, X.; Shao, G. Improved Artificial Potential Field based Parallel RRT Star for Fast Path Planning. In Proceedings of the 2021 China Automation Congress (CAC), Beijing, China, 22–24 October 2021; pp. 5801–5806.
26. Huang, S. Path planning based on mixed algorithm of RRT and artificial potential field method. In Proceedings of the 2021 4th International Conference on Intelligent Robotics and Control Engineering (IRCE), Lanzhou, China, 18–20 September 2021; pp. 149–155.
27. Yang, D.; Dong, L.; Dai, J.K. Collision avoidance trajectory planning for a dual-robot system: Using a modified APF method. *Robotica* **2024**, *42*, 846–863.
28. Luo, J.; Wang, Z.X.; Pan, K.L. Reliable path planning algorithm based on improved artificial potential field method. *IEEE Access* **2022**, *10*, 108276–108284.
29. Fan, J.; Chen, X.; Liang, X. UAV trajectory planning based on bi-directional APF-RRT* algorithm with goal-biased. *Expert Syst. Appl.* **2023**, *213*, 119137.
30. Yin, X.; Dong, W.; Wang, X.; Yu, Y.; Yao, D. Route planning of mobile robot based on improved RRT star and TEB algorithm. *Sci. Rep.* **2024**, *14*, 8942.
31. Yang, C.; Chen, Q.L.; Yan, X.F. An Overview and Comparison of Traditional Motion Planning Based on Rapidly Exploring Random Trees. *Sensors* **2025**, *25*, 2067.
32. An, B.; Kim, J.; Park, F.C. An adaptive stepsize RRT planning algorithm for open-chain robots. *IEEE Robot. Autom. Lett.* **2017**, *3*, 312–319.
33. Wang, J.; Chi, W.; Li, C.; Meng, M.Q. Efficient robot motion planning using bidirectional-unidirectional RRT extend function. *IEEE Trans. Autom. Sci. Eng.* **2021**, *19*, 1859–1868.
34. Cui, X.; Wang, C.; Xiong, Y.; Mei, L.; Wu, S. More Quickly-RRT*: Improved Quick Rapidly-exploring Random Tree Star algorithm based on optimized sampling point with better initial solution and convergence rate. *Eng. Appl. Artif. Intell.* **2024**, *133*, 108246.
35. Xu, T. Recent advances in Rapidly-exploring random tree: A review. *Heliyon* **2024**, *10*, e32451.
36. Zhang, R.; Guo, H.; Andriukaitis, D.; Li, Y.; Królczyk, G.; Li, Z. Intelligent path planning by an improved RRT algorithm with dual grid map. *Alex. Eng. J.* **2024**, *88*, 91–104.
37. Jiang, X.; Wang, Z.; Dong, C. A Path Planning Algorithm Based on Improved RRT Sampling Region. *Comput. Mater. Contin.* **2024**, *80*, 4303–4323.
38. Ou, Y.; Fan, Y.; Zhang, X.; Lin, Y.; Yang, W. Improved A* path planning method based on the grid map. *Sensors* **2022**, *22*, 6198.
39. Guo, S.; Zhang, X.; Du, Y.; Zheng, Y.; Cao, Z. Path planning of coastal ships based on optimized DQN reward function. *J. Mar. Sci. Eng.* **2021**, *9*, 210.
40. De Boor, C. *A Practical Guide to Splines*; Springer: New York, NY, USA, 1978; pp. 45–52.

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.