

Article

A Multi-Strategy Improved RRT Algorithm for Robot Path Planning

Jin Liu and Biao Leng *

Department of Civil Engineering, Southwest Jiaotong University, Chengdu 611756, China; liujin012005@163.com

* Correspondence: bleng@swjtu.cn; Tel.: +86-159-28163877

Abstract

To address the limitations of the traditional rapidly exploring random tree (RRT) algorithm, including redundant exploration, limited adaptability in dense environments, insufficient obstacle clearance, and poor path smoothness, this paper proposes an integrated multi-strategy improved RRT framework for robot path planning. The proposed method combines KD-Tree accelerated nearest-neighbor search, adaptive step-size adjustment, safety-boundary-based collision checking, direct goal-connection, safety-validated shortcut pruning, and spline-based path smoothing. Comparative experiments were conducted in four representative scenarios, including simple sparse, complex dense, narrow-passage, and highly cluttered environments. Traditional RRT, RRT*, RRT-Connect, and the proposed method were evaluated through 30 independent runs in each scenario. The results show that all algorithms achieved a 100% success rate in the tested scenarios. RRT* generated near-optimal paths but required substantially longer planning time and a much larger search tree, while RRT-Connect achieved the shortest planning time but provided relatively small obstacle clearance. In contrast, the proposed method achieved a better balance among path length, tree compactness, obstacle clearance, geometric smoothness, and computational efficiency. Ablation experiments further verified the contribution of each module, and parameter sensitivity analysis demonstrated the influence of the balancing factor, safety margin, and spline sampling number. The proposed framework provides a practical and lightweight path-planning solution for static two-dimensional environments.

Keywords: path planning; rapidly exploring random tree (RRT); KD-Tree; adaptive step size; node pruning; spline curve optimization

1. Introduction

Path planning is a fundamental technology for autonomous motion in intelligent systems such as mobile robots, autonomous vehicles, unmanned aerial vehicles (UAVs), and robotic manipulators. Its main objective is to generate a feasible and safe path from a given start point to a target point while avoiding obstacles and satisfying requirements related to path length, computational efficiency, motion smoothness, and operational safety [1–3]. With the increasing application of intelligent equipment in logistics, service robots, agriculture, inspection, and transportation scenarios, path-planning algorithms are required to operate reliably in environments that are not only large-scale but also dense, cluttered, and geometrically constrained [4,5]. In such scenarios, the planning algorithm must balance global exploration capability, local obstacle-avoidance accuracy, real-time performance, and path quality.



Academic Editor: Daniele Landi

Received: 15 May 2026

Revised: 10 June 2026

Accepted: 12 June 2026

Published: 18 June 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](https://creativecommons.org/licenses/by/4.0/)[Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

Existing path-planning methods can generally be divided into graph-search-based methods, artificial-potential-field-based methods, intelligent optimization methods, and sampling-based methods. Classical graph-search algorithms such as Dijkstra and A* can obtain feasible paths in structured environments, but they usually require map discretization and may suffer from high computational costs when the search space becomes large or high-dimensional [6]. The artificial potential field method has a simple structure and strong real-time performance, but it is prone to local minima in complex obstacle environments [7]. Intelligent optimization and learning-based methods can improve global search capability and adaptability, but their performance often depends on parameter tuning, training data, or computational resources [8,9]. Compared with these methods, sampling-based algorithms do not require explicit discretization of the entire configuration space and have shown strong adaptability to complex and high-dimensional path-planning problems [10].

Among sampling-based methods, the rapidly exploring random tree (RRT) algorithm has been widely used because of its simple implementation, probabilistic completeness, and strong exploration ability in unknown or complex spaces [10,11]. RRT incrementally builds a random search tree from the start point by sampling random points in the configuration space and extending the tree toward them until the target region is reached. This mechanism enables the algorithm to explore the free space efficiently without constructing a complete global map in advance. Therefore, RRT and its variants have been widely applied to mobile robot navigation, robotic manipulator motion planning, UAV path planning, and autonomous driving scenarios [4,5,12].

However, the traditional RRT algorithm still has several limitations that restrict its practical application. First, the use of a fixed expansion step size makes it difficult to adapt to different environmental conditions. A large step size can improve exploration efficiency in open areas but may cause collisions or miss feasible passages in dense obstacle regions, whereas a small step size improves local accuracy but generates excessive redundant nodes and increases planning time. Second, the random sampling mechanism may lead to blind exploration, especially in narrow passages, where the probability of generating useful samples is low. Third, the path generated by the traditional RRT algorithm is usually composed of multiple polyline segments with redundant nodes and sharp turning angles, resulting in poor geometric smoothness and difficulty in subsequent trajectory tracking. Finally, nearest-neighbor search and collision detection are commonly implemented through global traversal, resulting in increasing computational costs as the number of tree nodes and obstacles grows [13,14].

To overcome these limitations, many improved RRT-based algorithms have been proposed. Kuffner and LaValle proposed RRT-Connect, which constructs two rapidly exploring random trees from the start and goal configurations to improve single-query planning efficiency [15]. Karaman and Frazzoli proposed RRT*, which introduces rewiring and cost optimization mechanisms to achieve asymptotic optimality, but this improvement usually increases computational cost and convergence time [16]. Islam et al. proposed RRT*-Smart to accelerate the convergence of RRT* by combining path optimization and biased sampling strategies [17]. In addition, some studies have focused on directional guidance, path smoothing, optimized sampling, nearest-neighbor acceleration, and post-processing optimization to improve search efficiency and path quality [4,8,12,18,19]. For example, Fast-RRT improves planning speed and path optimality, highlighting the weaknesses of traditional RRT in search-time stability, narrow-channel performance, and path optimality [14].

In addition to algorithmic improvements, recent studies have emphasized that path planning should also be considered together with robot maneuverability and motion characteristics. Dolgov et al. investigated path planning for autonomous vehicles in

unknown semi-structured environments and highlighted the importance of planning robustness, smoothness, and environmental uncertainty in practical autonomous driving scenarios [20]. Cosenza et al. analyzed the maneuverability of a skid-steering six-wheeled robot using multibody modeling, showing that the planned trajectory may be affected by the motion characteristics and dynamic behavior of the robotic platform [21]. These studies indicate that geometric path planning should not only focus on path length and planning time, but should also consider path smoothness and the feasibility of subsequent trajectory tracking.

More recent RRT-based studies have further focused on adaptive sampling, guided expansion, safety-aware planning, and path post-processing. Li and Tong proposed an improved RRT method with adjustable sampling probability and sampling area to enhance mobile robot path-planning efficiency [22]. Oh et al. developed TA-RRT*, which uses terrain analysis to guide adaptive sampling and step-size adjustment, demonstrating the importance of environment-aware sampling in complex terrains [23]. Faroni et al. proposed an adaptive hybrid local–global sampling strategy for informed sampling-based optimal planning, showing that dynamically balancing exploration and exploitation can improve convergence performance [24]. Huang et al. proposed a UAV path-planning algorithm based on an improved artificial potential field and bidirectional RRT*, integrating adaptive sampling and dynamic step-size adjustment in complex three-dimensional environments [25]. Fan et al. introduced BI-RRT*, which combines bidirectional search, obstacle expansion, informed sampling, and path refinement to improve path safety and planning performance [26]. Lei et al. proposed a multi-strategy fusion RRT algorithm that integrates target bias, adaptive expansion, path optimization, and smoothing strategies [27]. These studies show that recent RRT improvements are increasingly moving from single-module optimization toward integrated frameworks that jointly consider search efficiency, obstacle clearance, path compactness, and geometric smoothness.

Although the above studies have improved RRT-based planning from different perspectives, several issues remain unresolved. First, many methods mainly focus on a single aspect, such as sampling efficiency, path shortening, or path smoothing, while the coordination among adaptive exploration, collision safety, tree compactness, and path smoothness remains insufficient. Second, some algorithms improve path quality but introduce additional computational cost, limiting their suitability for lightweight real-time applications. Third, only limited studies provide module-wise ablation analysis and parameter sensitivity analysis to explain how each strategy contributes to the overall performance. Therefore, it is necessary to develop and evaluate an integrated RRT improvement framework that balances search efficiency, obstacle clearance, path compactness, and geometric smoothness.

To address the above problems, this paper proposes an integrated multi-strategy improved RRT framework for robot path planning. Rather than proposing a single isolated modification, this study combines several complementary modules within a unified implementation, including KD-Tree acceleration, adaptive step-size adjustment, safety-boundary collision checking, direct goal-connection, shortcut pruning, and spline-based path smoothing. The main contributions of this study are summarized as follows:

- (1) A lightweight integrated RRT improvement framework is developed by combining KD-Tree acceleration, adaptive step-size adjustment, safety-boundary collision checking, direct goal-connection, shortcut pruning, and spline-based path smoothing.
- (2) A direct goal-connection attempt and safety-validated shortcut pruning strategy are incorporated to reduce redundant tree expansion and remove unnecessary intermediate path nodes while maintaining collision-free feasibility.

- (3) Comparative experiments are conducted against traditional RRT, RRT*, and RRT-Connect in four different scenarios, including simple sparse, complex dense, narrow-passage, and highly cluttered environments.
- (4) A module-wise ablation study is performed to evaluate the individual contributions of KD-Tree acceleration, adaptive step-size adjustment, safety-boundary checking, pruning, and smoothing.
- (5) A parameter sensitivity analysis is conducted for the balancing factor, safety margin, and spline sampling number. In addition, path smoothness is evaluated using turning-angle-based metrics to avoid overstating kinematic feasibility in a purely geometric planning framework.

The remainder of this paper is organized as follows. Section 2 introduces the principle and limitations of the traditional RRT algorithm. Section 3 presents the proposed multi-strategy improved RRT framework, including adaptive step-size adjustment, enhanced collision detection, KD-Tree acceleration, goal-connection, path pruning, smoothing, and computational complexity analysis. Section 4 describes the simulation experiments and analyzes the comparative results, ablation study, and parameter sensitivity analysis under different scenarios. Section 5 summarizes the main conclusions and discusses future research directions.

2. Principle of the Traditional RRT Algorithm

Before introducing the proposed improved RRT framework, this section first describes the basic principle and implementation process of the traditional RRT algorithm. The traditional RRT algorithm expands a random search tree from the start point by repeatedly performing random sampling, nearest-neighbor search, node extension, collision detection, and goal-region checking. By analyzing these core procedures, the limitations of the traditional algorithm can be clarified, providing a theoretical basis for the subsequent improvements in adaptive step-size adjustment, KD-Tree acceleration, enhanced collision detection, and path smoothing.

(1) Determining the core parameters and initial state

The path-planning environment is defined as a two-dimensional map with boundary X_{map} . The start point and goal point are denoted as q_{start} and q_{goal} , respectively. The obstacle space is represented by X_{obs} , and the free space is denoted as $X_{free} = X_{map} \setminus X_{obs}$. The random tree is defined as $T = (V, E)$, where V is the node set and E is the edge set. At initialization, the start point q_{start} is added to V as the root node, and the edge set E is empty. The main algorithmic parameters include the maximum number of iterations N_{max} , the fixed expansion step size η , the goal-region threshold d_{goal} , and the goal-biased sampling probability p_{goal} .

(2) Generating candidate nodes for tree expansion

At each iteration, a candidate sampling point q_{rand} is generated to guide the expansion of the random tree. To improve convergence toward the goal, the traditional RRT algorithm may adopt a goal-biased sampling strategy. With a probability of p_{goal} , the goal point q_{goal} is selected as the sampling point; otherwise, a random point is uniformly sampled from the free space. This process can be expressed as

$$q_{rand} = \begin{cases} q_{goal}, & \text{with probability } p_{goal} \\ \text{Random}(X_{free}), & \text{with probability } 1 - p_{goal} \end{cases} \quad (1)$$

(3) Determining the base point for tree expansion

After obtaining q_{rand} , the algorithm searches the existing node set V to find the node closest to q_{rand} . For any two nodes $q_i = (x_i, y_i)$ and $q_j = (x_j, y_j)$, the Euclidean distance is defined as

$$d(q_i, q_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}, \quad (2)$$

Based on this distance metric, the nearest node q_{near} is selected as

$$q_{near} = \underset{q_i \in V}{\operatorname{argmin}} d(q_i, q_{rand}), \quad (3)$$

In the traditional RRT algorithm, this nearest-neighbor search is usually implemented by linear traversal of all existing tree nodes, which leads to a time complexity of $O(n)$, where n is the number of nodes in the tree.

(4) Constructing a new branch of the tree

Starting from q_{near} , a new node q_{new} is generated by extending toward q_{rand} with a fixed step size η . If the distance between q_{near} and q_{rand} is smaller than η , q_{rand} is directly taken as q_{new} . Otherwise, q_{new} is calculated as

$$q_{new} = q_{near} + \eta \frac{q_{rand} - q_{near}}{\|q_{rand} - q_{near}\|}, \quad (4)$$

This fixed-step expansion mechanism is simple to implement, but it lacks adaptability in different environmental conditions. A large step size may cause collisions or miss narrow passages, whereas a small step size may generate excessive redundant nodes and increase planning time.

(5) Verifying node and path safety

After q_{new} is generated, collision detection is performed to determine whether the new node and the connecting edge are located in the free space. The collision detection process includes point collision checking and line-segment collision checking. First, the algorithm checks whether q_{new} lies inside any obstacle. For a circular obstacle with center q_{obs} and radius r_{obs} , a point collision occurs if the distance between q_{new} and q_{obs} is less than or equal to r_{obs} . Second, the line segment $\overline{q_{near}q_{new}}$ is checked to determine whether it intersects any obstacle. If either the point collision check or the line-segment collision check fails, q_{new} is discarded. Otherwise, q_{new} is added to the node set V , the edge (q_{near}, q_{new}) is added to E , and q_{near} is recorded as the parent node of q_{new} .

(6) Detecting whether the goal region has been reached

After the new node is added to the random tree, the algorithm checks whether it has reached the goal region. If the distance between q_{new} and q_{goal} is smaller than the goal threshold d_{goal} , and the line segment from q_{new} to q_{goal} is collision-free, the path is considered successfully found. The final path is then obtained by backtracking from the goal node to the root node q_{start} through the parent-node chain. If no feasible path is found after N_{max} iterations, the planning process is considered unsuccessful.

The principle of the traditional RRT algorithm is illustrated in Figure 1. Through continuous sampling and tree expansion, the algorithm gradually explores the free space and attempts to construct a feasible path from q_{start} to q_{goal} .

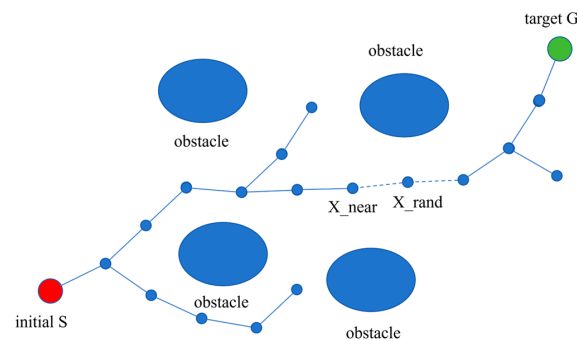


Figure 1. Schematic diagram of the RRT algorithm.

3. Improved RRT Algorithm Design

3.1. Adaptive Step-Size Adjustment Strategy Design

The traditional RRT algorithm uses a fixed step size to expand the search tree, which limits its adaptability in complex environments. In open areas with sparse obstacles, a fixed step size may restrict the exploration range in each iteration, prolonging the time required for the search tree to cover the free space and generating redundant nodes. In contrast, in regions close to the target or in narrow passages with dense obstacles, the same fixed step size may lack sufficient precision, causing the expansion to miss feasible connections or collide with obstacles. Consequently, the algorithm may fail to find a feasible path or may generate a non-optimal path.

To address this issue, this paper proposes an adaptive step-size strategy that integrates environmental perception and goal orientation. The step size is dynamically adjusted according to the local environmental complexity around the current expansion node and its proximity to the target. This strategy allows the algorithm to use larger steps for efficient exploration in open areas, smaller steps for precise obstacle avoidance in complex regions, and goal-oriented adjustment for faster convergence near the target region. In this way, the method improves exploration efficiency while maintaining obstacle-avoidance safety.

To support stable and interpretable step-size adjustment, the proposed strategy first defines distance-related parameters that describe the local obstacle distribution and the relative position of the target. These parameters provide quantitative inputs for step-size decision-making and reduce the uncertainty caused by purely empirical adjustments. Two normalized weighting factors are then constructed to represent obstacle-safety constraints and target-convergence requirements. Finally, these factors are combined through a balancing factor and bounded by upper and lower step-size limits to generate a step size suitable for the current expansion state.

First, let the current expansion node be $N_{near}(x_{near}, y_{near})$. Using KD-Tree, the neighboring obstacle set Obs_{near} within a range of $5r_{obs}$ (where r_{obs} denotes the obstacle radius) is rapidly screened. The minimum Euclidean distance D_{obs} from the node to this obstacle set is calculated. This parameter directly reflects the safety margin between the node and surrounding obstacles. The smaller D_{obs} is, the more likely the node is located in an obstacle-dense region with higher environmental complexity, and thus the step size should be reduced to decrease collision risk and ensure safe node expansion. Conversely, the larger D_{obs} is, the more likely the node lies in an open region, allowing the step size to be increased to improve exploration efficiency. The mathematical expression is as follows:

$$D_{obs} = \min_{(x_o, y_o) \in Obs_{near}} \sqrt{(x_{near} - x_o)^2 + (y_{near} - y_o)^2}, \quad (5)$$

Second, let the starting point be $S(x_s, y_s)$ and the goal point be $G(x_g, y_g)$. The straight-line distance from the current node to the goal point is defined as D_{goal} using the Euclidean

distance formula. The initial distance between the start and goal points is $D_{goal,init}$. Using this reference distance, D_{goal} is converted into a relative value, so that the calculation of the goal-oriented factor is not affected by the map size, thereby improving the adaptability of the step-size strategy.

Based on these distance parameters, two normalized weighting factors are constructed.

The first is the goal bias factor f_{goal} , which characterizes the degree to which a node approaches the goal. The larger the factor is, the stronger the adaptability of the step size toward convergence to the goal. The factor is constrained within the interval $[0, 1]$, where $f_{goal} \approx 1$ near the goal and $f_{goal} \approx 0$ near the starting point. Its mathematical expression is as follows:

$$f_{goal} = 1 - \min\left(\frac{D_{goal}}{D_{goal,init}}, 1.0\right), \quad (6)$$

The second is the obstacle-safety factor f_{obs} , which represents the safety margin between the node and obstacles. The larger the factor is, the wider the adjustable range of the step size. Meanwhile, a minimum threshold is set to prevent the step size from becoming too small, which may lead to exploration stagnation. Its mathematical expression is as follows:

$$f_{obs} = \min\left(\frac{D_{obs}}{4r_{obs}}, 1.0\right), f_{obs} = \max(f_{obs}, 0.3), \quad (7)$$

where $4r_{obs}$ is an empirical safety threshold verified through repeated experiments. When $D_{obs} \geq 4r_{obs}$, the node is located in a completely safe open region, and obstacles impose no constraint on the step size, thus the factor is set to 1. When $D_{obs} < 4r_{obs}$, the factor decreases as D_{obs} decreases, and the step size correspondingly becomes smaller. Meanwhile, the $\max$ function sets the lower bound of the factor to 0.3, ensuring that even in extremely complex obstacle regions, the step size will not be so excessively small as to cause exploration stagnation, thereby maintaining basic exploration efficiency.

Based on the above two normalized factors, a dynamic step-size calculation model is constructed. The model is expressed as follows:

$$step = step_{min} + (step_{max} - step_{min}) \cdot (\alpha \cdot f_{obs} + (1 - \alpha) \cdot f_{goal}), \quad (8)$$

where $step_{min} = 0.2$ and $step_{max} = 1.0$ represent the lower and upper bounds of the step size, respectively, and $\alpha = 0.6$ is the balancing factor used to balance obstacle-avoidance safety and convergence speed toward the goal.

To accelerate convergence near the goal region, a step-size amplification rule is introduced. When the current node is close to the goal, the step size is multiplied by 1.1. The resulting value is then clipped to the predefined range to avoid excessively small steps that may cause stagnation or excessively large steps that may increase collision risk.

The adaptive step size is illustrated in Figure 2, where the step size is larger in open regions and smaller in dense regions.

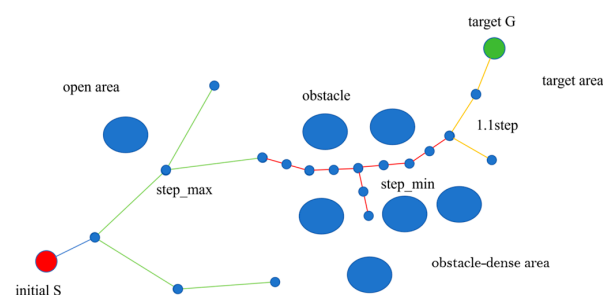


Figure 2. Principle of adaptive step size.

3.2. Enhanced Collision Detection and Safety-Boundary Design

Collision detection is the core guarantee of path-planning safety. Traditional detection methods suffer from insufficient accuracy and low efficiency. In this study, a triple detection mechanism of “node collision + line segment collision + safety boundary” is designed. Combined with KD-Tree acceleration, the proposed method improves detection accuracy while ensuring the real-time performance of the algorithm.

Based on the original obstacle radius r_{obs} , a safety margin of 0.2 is added to form a safety radius $r_{safe} = r_{obs} + 0.2$. In the proposed method, the safety radius is used as the reference for enhanced collision checking to reduce the risk of paths passing too close to obstacles.

To determine whether a node lies within the safety boundary, let the node to be tested be $P(x, y)$. Using KD-Tree, the obstacle set within a radius of $3r_{safe}$ is rapidly queried. The Euclidean distance d between the node and the center of each obstacle is calculated. If there exists an obstacle such that $d \leq r_{safe}$, the node is determined to be in collision and is directly discarded.

To verify whether the line segment P_1P_2 intersects with the obstacle-safety boundary, a lightweight detection strategy of “midpoint pre-check + dynamic sampling detection” is adopted. First, the midpoint of the line segment is calculated as $P_{mid}(\frac{x_1+x_2}{2}, \frac{y_1+y_2}{2})$. If the midpoint collides with an obstacle, the line segment is directly judged to be in collision without performing subsequent sampling. Otherwise, the sampling density is adaptively adjusted according to the length of the line segment. The longer the segment, the greater the number of sampling points (with a minimum sampling number of 5). All sampling points are traversed to perform collision verification, ensuring that the line segment has no collision risk.

KD-Tree reduces the cost of local obstacle querying from linear traversal to approximately logarithmic search, thereby decreasing the number of unnecessary distance calculations during collision detection.

As shown in Figure 3, the collision detection efficiency is enhanced by introducing the safety margin.

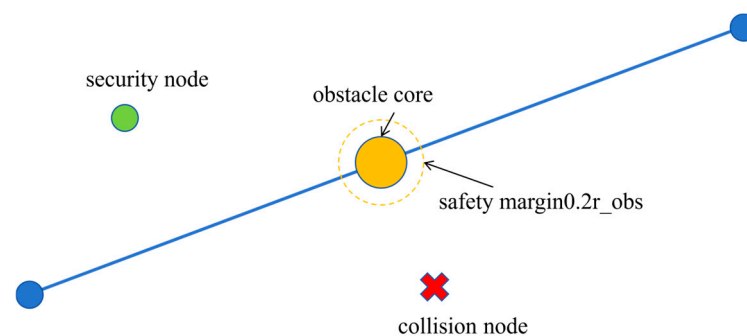


Figure 3. Principle of enhanced collision detection and safety boundary.

3.3. Path Simplification and Smoothing Design

The original path generated by the RRT algorithm usually consists of a series of polyline segments and presents two main shortcomings. First, it may contain a large number of redundant intermediate nodes, which increases the path length and reduces execution efficiency. Second, the path often contains sharp turning angles, resulting in poor geometric smoothness and making subsequent trajectory tracking more difficult. To address these problems, this study designs a post-processing framework that combines greedy path simplification, safety-validated shortcut pruning, and spline-based smoothing.

The purpose of this framework is to reduce redundant nodes, improve path compactness, and generate a smoother geometric reference path.

The first stage is greedy path simplification. Let the original path node sequence be $P = \{q_1, q_2, \dots, q_p\}$, where $q_1 = q_{start}$ and $q_p = q_{goal}$. Starting from the current node q_i , the algorithm searches forward along the path and selects the farthest node q_j that can be directly connected to q_i without collision. All intermediate nodes between q_i and q_j are removed. This process is repeated until the goal node is reached. Compared with iterative simplification methods, this greedy strategy is lightweight and suitable for real-time path planning. To ensure safety, each shortcut segment must pass the line-segment collision detection before intermediate nodes are removed.

In addition to greedy simplification, a safety-validated shortcut pruning step is further applied. Two non-adjacent nodes in the path are selected, and the intermediate nodes are removed only when the direct connection between them is collision-free. This additional pruning step further reduces unnecessary intermediate nodes while preserving path feasibility. If a shortcut connection fails the safety verification, the original path segment is retained. Therefore, the pruning process improves path compactness without sacrificing collision-free safety.

During tree expansion, a direct goal-connection attempt is also introduced. After a valid new node is generated, the algorithm attempts to directly connect this node to q_{goal} . If the line segment between the new node and the goal point is collision-free, the search process terminates early, and the final path is generated through parent-node backtracking. This strategy helps reduce unnecessary exploration when a feasible connection to the goal already exists, thereby improving search efficiency and reducing the scale of the random tree.

After path simplification and shortcut pruning, the resulting path may still contain polyline segments with abrupt turning angles. Therefore, spline-based smoothing is applied to improve geometric smoothness. To avoid path distortion caused by uneven spacing between adjacent path nodes, cumulative arc length is used as the spline parameter. Let the simplified path be $P_s = \{q_1, q_2, \dots, q_m\}$. The cumulative arc length is defined as

$$s_1 = 0, \quad s_i = \sum_{j=2}^i \|q_j - q_{j-1}\|, \quad i = 2, 3, \dots, m, \quad (9)$$

The normalized arc-length parameter is then calculated as

$$u_i = \frac{s_i}{s_m}, \quad i = 1, 2, \dots, m, \quad (10)$$

Based on the normalized parameter u_i , spline interpolation functions are constructed for the x - and y -coordinates:

$$x = f_x(u), \quad y = f_y(u), \quad u \in [0, 1], \quad (11)$$

In this study, 60 points are uniformly sampled over the normalized arc-length parameter to generate the smoothed path. This value is selected according to the parameter sensitivity analysis, which shows that 60 samples provide a better balance between geometric smoothness and computational cost compared with smaller sampling numbers.

To ensure that smoothing does not introduce unsafe path segments, a safety verification mechanism is applied after interpolation. All sampled points and adjacent line segments of the smoothed path are checked against the obstacle-safety boundary. If the smoothed path is collision-free and remains within the map boundary, it is accepted as the final path. Otherwise, the algorithm adopts a safety rollback strategy and returns to the

pruned path. In this way, the final output path maintains collision-free feasibility while improving geometric smoothness.

It should be noted that the proposed planner is evaluated in a two-dimensional geometric environment and does not explicitly model nonholonomic constraints, steering limits, or vehicle dynamics. Therefore, the smoothing process is used to improve geometric smoothness and provide a smoother reference path for subsequent trajectory tracking, rather than to guarantee full kinematic feasibility.

As shown in Figure 4, the red path represents the smoothed path. Blue dots indicate tree nodes.

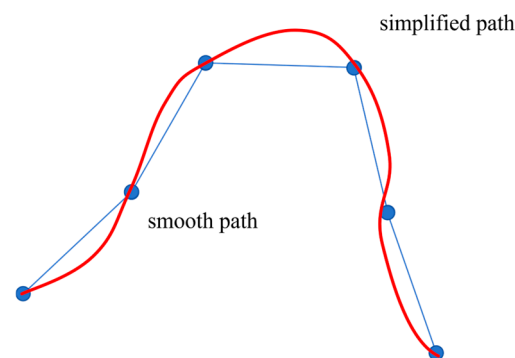


Figure 4. Path simplification and smoothing principles.

3.4. KD-Tree Accelerated Nearest-Neighbor Search Design

Nearest-neighbor search is a core component of the RRT algorithm. It is used to find the node in the search tree that is closest to the randomly sampled point, which then serves as the base node for the expansion of a new node. Traditional RRT algorithms adopt a global traversal method to find the nearest neighbor, with a time complexity of $O(n)$. As the number of nodes in the search tree increases, the traversal time grows linearly. In large-scale search scenarios, this approach becomes extremely inefficient and constitutes a major bottleneck for the real-time performance of the algorithm.

To address this issue, this paper introduces the KD-Tree data structure to construct an index for the random tree nodes. This reduces the complexity of nearest-neighbor search from $O(n)$ to $O(\log n)$, significantly improving the exploration efficiency of the algorithm. Furthermore, it works collaboratively with other modules to further enhance the overall system performance.

KD-Tree is not only used for nearest-neighbor search but also cooperates with the previously designed adaptive step-size strategy and enhanced collision detection mechanism, forming a globally optimized framework that further improves the overall operational efficiency of the algorithm.

First, in cooperation with the adaptive step-size strategy, during step-size calculation it is necessary to obtain the set of neighboring obstacles around the current node to compute the parameter D_{obs} . By using KD-Tree, the local obstacle set can be quickly filtered without globally traversing all obstacles. This greatly improves the computational efficiency of D_{obs} , providing rapid data support for dynamic step-size adjustment.

Second, in cooperation with the collision detection mechanism, both node collision detection and line-segment collision detection require screening obstacles near the object to be tested. By utilizing the nearest-neighbor query capability of KD-Tree, local obstacles can be quickly located, thereby reducing unnecessary distance calculations. This further decreases the computational cost of collision detection and improves the overall real-time performance of the algorithm.

3.5. Computational Complexity Analysis

To further analyze the computational efficiency of the proposed framework, the computational complexity of each main module is discussed in this section. Let n denote the number of random tree nodes, m denote the number of obstacles, k denote the number of local obstacles returned by the KD-Tree query, s denote the number of sampled points used for line-segment collision checking, p denote the number of path nodes before post-processing, and h denote the number of spline sampling points.

In the traditional RRT algorithm, nearest-neighbor search is usually implemented by linear traversal over all existing tree nodes, resulting in a time complexity of $O(n)$ for each iteration. In the proposed framework, KD-Tree is used to accelerate nearest-neighbor search, reducing the query complexity to approximately $O(\log n)$. Similarly, local obstacle querying is accelerated by KD-Tree, reducing the obstacle search complexity from $O(m)$ to approximately $O(\log m + k)$, where k is the number of local obstacles returned by the query.

The adaptive step-size calculation depends on the local obstacle distribution around the current expansion node. Since the local obstacle set is obtained through KD-Tree query, the complexity of this module is approximately $O(\log m + k)$. For line-segment collision checking, the traditional global traversal strategy requires checking all obstacles for each sampled point, resulting in a complexity of $O(s \cdot m)$. In contrast, the proposed method first queries local obstacles and then performs collision checking only within the local obstacle set, reducing the complexity to approximately $O(s \cdot k)$.

For path post-processing, greedy simplification and shortcut pruning remove redundant intermediate nodes through collision-free connection tests. In the worst case, the complexity of the pruning process is $O(p^2)$, where p is the number of path nodes. However, the actual computational cost is usually lower because early termination and safety pre-checking are used to avoid unnecessary collision checks. The spline smoothing module generates h sampled points along the normalized arc-length parameter, and its complexity is approximately $O(h)$.

The computational complexity of the main modules is summarized in Table 1.

Table 1. Computational complexity comparison of different modules.

Module	Traditional RRT	Proposed Method
Nearest-neighbor search	$O(n)$	$O(\log n)$
Local obstacle query	$O(m)$	$O(\log m + k)$
Adaptive step-size calculation	Not included	$O(\log m + k)$
Line-segment collision checking	$O(s \cdot m)$	$O(s \cdot k)$
Goal-connection attempt	Not included	$O(s \cdot k)$
Greedy and shortcut pruning	Not included	$O(p^2)$ in the worst case
Spline smoothing	Not included	$O(h)$

Overall, the proposed method introduces additional operations for adaptive step-size adjustment, safety-boundary checking, shortcut pruning, and spline smoothing. However, KD-Tree acceleration, direct goal-connection, and pruning strategies reduce redundant tree expansion and local search costs. Therefore, the proposed framework improves overall planning efficiency by reducing unnecessary node expansion and limiting collision checking to local obstacle regions.

3.6. Complete Procedure of the Improved RRT Algorithm

The improved RRT algorithm is built upon the core framework of the traditional RRT algorithm, namely “random sampling—tree expansion—path backtracking.” By integrating four key technologies—adaptive step-size adjustment, enhanced collision detection with safety boundaries, path simplification and smoothing, and KD-Tree accelerated nearest-neighbor search—a complete closed-loop framework of “intelligent exploration—path backtracking—post-processing optimization” is formed.

This approach not only addresses the problems of low efficiency, poor path quality, and insufficient safety in traditional algorithms but also ensures real-time performance and general applicability. The detailed procedure is as follows:

- Input core parameters: starting point S , goal point G , map boundary, obstacle set Obs , and obstacle radius r_{obs} .
- Construct auxiliary structures: build KD-Tree structures for both obstacles and random tree nodes, which are globally reused to improve query efficiency.
- Initialize the random tree: add the starting point as the root node of the tree and set its parent node as None.
- Configure hyperparameters: maximum iteration number $N_{max} = 5000$, goal distance threshold $d_{thresh} = 0.75$, and goal-biased sampling probability $\{p_{goal} = 0.2\}$.
- Random sampling: sample the goal point with a probability of 20%, and sample randomly within the map range with a probability of 80%, ensuring that the sampled point lies in free space.
- Nearest-neighbor search: use KD-Tree to rapidly locate the node N_{near} in the random tree that is closest to the sampled point.
- Adaptive step-size calculation: compute the step size using the dynamic model based on the local environment around N_{near} and its proximity to the goal.
- New node generation: generate a new node along the direction from N_{near} to the sampled point and clip its coordinates within the map boundary.
- Enhanced collision detection: verify the safety of the new node and the connecting segment using the triple detection mechanism of node collision + line segment collision + safety boundary.
- Goal check: if the distance between the new node and the goal is smaller than the threshold and the connecting segment is collision-free, the goal is considered reachable.
- Path backtracking: traverse the parent nodes backward from the goal node to obtain the original path and reverse its order.
- Path simplification: remove redundant nodes using the greedy strategy to obtain a simplified path.
- Path smoothing: apply cubic spline interpolation to optimize the path, and output the final path after collision verification.
- Termination condition: if a feasible path is found, output the result; if the iteration number reaches the maximum limit without finding a path, the problem is considered infeasible.

The improved RRT algorithm flowchart is shown in Figure 5.

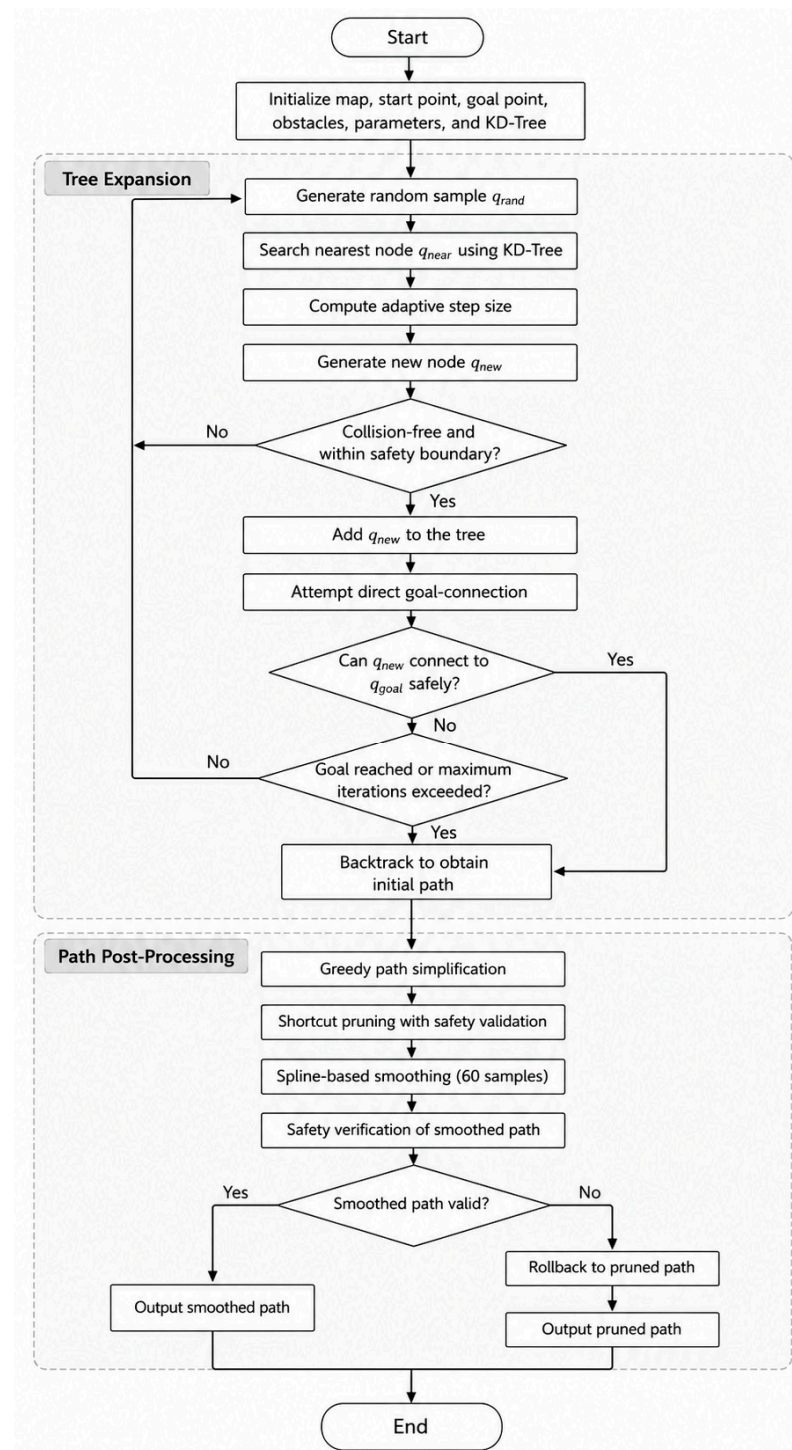


Figure 5. The improved RRT algorithm flowchart.

4. Simulation Experiments and Result Analysis

To evaluate the effectiveness of the proposed multi-strategy improved RRT framework, simulation experiments were conducted under different obstacle distributions and compared with existing RRT-based algorithms. The experimental design was revised to address the randomness of RRT-based planners and to provide a more comprehensive validation. Specifically, traditional RRT, RRT*, RRT-Connect, and the proposed method were compared under four scenarios, and each algorithm was independently executed 30 times using different random seeds. In addition to the main comparison, ablation experiments and

parameter sensitivity analysis were conducted to evaluate the contribution of each module and the influence of key parameters.

4.1. Experimental Software and Hardware Environment

The simulation experiments were implemented in a Python environment. Python 3.8 or above was adopted, with core libraries including NumPy 1.21.0, Matplotlib 3.4.0, SciPy 1.7.0, and Pandas 1.3.0. NumPy was used for numerical computation, SciPy was used for KD-Tree construction and spline interpolation, Matplotlib was used for visualization, and Pandas was used for data recording and statistical analysis.

The experiments were conducted on a computer equipped with an Intel Core i5-10400F CPU and 16 GB RAM, running the Windows 10 64-bit operating system. GPU acceleration was not used. Therefore, all algorithms were evaluated under the same computational resource conditions, which ensured a fair comparison of planning time and computational efficiency.

4.2. Parameter Configuration

To ensure fairness, all algorithms were tested under the same map range, start and goal coordinates, obstacle settings, and maximum iteration number. Algorithm-specific parameters were configured according to the corresponding algorithmic mechanisms. For example, traditional RRT uses a fixed step length, while the proposed method uses adaptive step-size adjustment, safety-boundary checking, KD-Tree acceleration, goal-connection, shortcut pruning, and spline-based smoothing. The core parameter settings are shown in Table 2.

Table 2. Core algorithm parameter configuration.

Parameter Name	Values	Parameter Description
Map range	(0, 20, 0, 20)	Range of the X and Y axes in the 2D plane
Start coordinates	(1, 1)	Path-planning start point
Goal coordinates	(19, 19)	Path-planning target point
Obstacle radius	0.3	Original geometric radius of each circular obstacle
Safety margin	0.2	Additional safety buffer used in the proposed collision checking
Goal distance threshold	0.75	Distance threshold for determining whether the goal region is reached
Maximum number of iterations	5000	Termination condition for avoiding infinite iterations
Traditional RRT fixed step length	0.5	Fixed expansion step length of the traditional RRT algorithm
Improved RRT step-size range	(0.2, 1.0)	Minimum and maximum values of adaptive step size
Balancing factor α	0.6	Balance between obstacle avoidance and goal convergence
Goal-biased sampling probability	0.2	Probability of sampling the goal point
Path smoothing sampling count	60	Number of spline sampling points, selected according to parameter sensitivity analysis

4.3. Experimental Scenarios and Evaluation Metrics

Four representative two-dimensional static scenarios were designed to evaluate planning performance under different obstacle distributions. The simple sparse scenario contains sparsely distributed obstacles and is used to evaluate basic exploration efficiency. The complex dense scenario contains more densely distributed obstacles and is used to test local obstacle-avoidance ability. The narrow-passage scenario contains constrained passages and is used to evaluate performance in geometrically restricted environments. The highly cluttered scenario contains a larger number of obstacles and is used to further examine the effect of redundant exploration and safety-aware planning in more complex obstacle distributions.

The compared algorithms include traditional RRT, RRT*, RRT-Connect, and the proposed method. Traditional RRT refers to a fixed-step RRT algorithm with linear nearest-neighbor search and basic collision checking. RRT* and RRT-Connect were included as stronger RRT-based baselines to provide a more rigorous comparison. The proposed method refers to the integrated framework combining KD-Tree acceleration, adaptive step-size adjustment, safety-boundary checking, goal-connection, shortcut pruning, and spline-based smoothing.

To account for the inherent randomness of RRT-based algorithms, each algorithm was independently executed 30 times in each scenario using different random seeds. The planning success rate was calculated over all trials, while path length, planning time, tree nodes, minimum obstacle clearance, maximum turning angle, mean turning angle, and turning-angle standard deviation were reported as mean $\pm$ standard deviation over successful runs. The minimum obstacle clearance was used to evaluate how close the planned path was to obstacles. Turning-angle-based metrics were used to evaluate geometric smoothness. Since the current planner is evaluated in a two-dimensional geometric environment, these metrics are used to describe geometric smoothness rather than full dynamic or kinematic feasibility.

4.4. Comparison with Existing RRT-Based Algorithms

Table 3 presents the main comparison results. All four algorithms achieved a 100% success rate in the tested scenarios. Therefore, the performance differences are mainly reflected in tree compactness, obstacle clearance, geometric smoothness, and computational efficiency rather than in planning success rate.

Table 3 shows that RRT* generated near-optimal paths in most scenarios, but this improvement was achieved at the cost of substantially longer planning time and a much larger search tree. For example, the average planning time of RRT* was approximately 24 s, and the number of tree nodes exceeded 4500 in all tested scenarios. RRT-Connect achieved the shortest planning time because of its bidirectional expansion mechanism; however, its minimum obstacle clearance was relatively small, especially in the complex dense and narrow-passage scenarios.

Compared with traditional RRT, the proposed method substantially reduced redundant tree expansion. The average tree-node number decreased from 257.93 to 19.47 in the simple sparse scenario, from 268.50 to 42.10 in the complex dense scenario, from 266.83 to 85.63 in the narrow-passage scenario, and from 284.00 to 4.43 in the highly cluttered scenario. In addition, the proposed method achieved the largest minimum obstacle clearance in all tested scenarios, indicating that the safety-boundary checking mechanism enhanced path safety.

Table 3. Performance comparison of different RRT-based algorithms. Values are reported as mean $\pm$ standard deviation over 30 independent runs.

Scenario	Algorithm	Success Rate (%)	Path Length	Time (s)	Tree Nodes	Min Clearance	Mean Angle (°)
Simple sparse	Traditional RRT	100	30.72 $\pm$ 1.02	0.046 $\pm$ 0.017	257.93 $\pm$ 55.87	0.157 $\pm$ 0.115	147.30 $\pm$ 3.12
Simple sparse	RRT*	100	25.58 $\pm$ 0.07	24.993 $\pm$ 2.960	4909.97 $\pm$ 12.67	0.170 $\pm$ 0.113	174.97 $\pm$ 0.86
Simple sparse	RRT-Connect	100	26.50 $\pm$ 1.09	0.005 $\pm$ 0.001	58.33 $\pm$ 6.18	0.081 $\pm$ 0.080	174.00 $\pm$ 2.67
Simple sparse	Proposed	100	26.02 $\pm$ 0.63	0.061 $\pm$ 0.010	19.47 $\pm$ 9.87	0.600 $\pm$ 0.332	176.34 $\pm$ 8.75
Complex dense	Traditional RRT	100	30.25 $\pm$ 1.22	0.056 $\pm$ 0.020	268.50 $\pm$ 56.23	0.080 $\pm$ 0.052	148.62 $\pm$ 3.63
Complex dense	RRT*	100	25.60 $\pm$ 0.05	24.230 $\pm$ 0.202	4792.90 $\pm$ 16.39	0.156 $\pm$ 0.094	174.58 $\pm$ 1.62
Complex dense	RRT-Connect	100	26.51 $\pm$ 0.53	0.006 $\pm$ 0.001	58.93 $\pm$ 5.95	0.045 $\pm$ 0.034	171.92 $\pm$ 3.17
Complex dense	Proposed	100	26.78 $\pm$ 1.32	0.111 $\pm$ 0.034	42.10 $\pm$ 18.99	0.469 $\pm$ 0.326	156.10 $\pm$ 29.50
Narrow passage	Traditional RRT	100	30.65 $\pm$ 1.20	0.059 $\pm$ 0.023	266.83 $\pm$ 58.91	0.043 $\pm$ 0.031	146.83 $\pm$ 2.70
Narrow passage	RRT*	100	25.58 $\pm$ 0.04	24.100 $\pm$ 0.250	4709.77 $\pm$ 23.39	0.069 $\pm$ 0.042	174.47 $\pm$ 1.10
Narrow passage	RRT-Connect	100	26.50 $\pm$ 0.54	0.006 $\pm$ 0.001	58.23 $\pm$ 4.52	0.045 $\pm$ 0.031	172.67 $\pm$ 2.63
Narrow passage	Proposed	100	32.79 $\pm$ 1.24	0.121 $\pm$ 0.018	85.63 $\pm$ 19.21	1.115 $\pm$ 0.575	176.89 $\pm$ 7.51
Highly cluttered	Traditional RRT	100	31.46 $\pm$ 2.07	0.078 $\pm$ 0.063	284.00 $\pm$ 106.67	0.061 $\pm$ 0.047	147.43 $\pm$ 3.23
Highly cluttered	RRT*	100	25.54 $\pm$ 0.03	24.393 $\pm$ 0.284	4591.70 $\pm$ 28.89	0.329 $\pm$ 0.156	175.26 $\pm$ 1.64
Highly cluttered	RRT-Connect	100	25.50 $\pm$ 0.05	0.007 $\pm$ 0.001	52.37 $\pm$ 0.49	0.295 $\pm$ 0.112	179.60 $\pm$ 0.28
Highly cluttered	Proposed	100	25.46 $\pm$ 0.00	0.022 $\pm$ 0.005	4.43 $\pm$ 3.90	2.834 $\pm$ 0.596	180.00 $\pm$ 0.00

It should be noted that some metrics exhibit very low variance in the highly cluttered scenario, especially for the proposed method. As shown in Table 3, the proposed method obtained a path length of 25.46 ± 0.00 and a mean turning angle of 180.00 ± 0.00 in this scenario. This low variance is mainly related to the geometric structure of the tested map and the post-processing mechanism of the proposed framework. In the highly cluttered environment, once the tree expansion reaches a feasible connection region, the direct goal-connection, safety-validated shortcut pruning, and spline-based smoothing steps tend to transform different initial random trees into a similar final geometric path. Therefore, the final path length and mean turning angle may become nearly identical across repeated runs. However, the planning time, tree-node number, and minimum obstacle clearance still show variability, with values of 0.022 ± 0.005 s, 4.43 ± 3.90 , and 2.834 ± 0.596 , respectively. This indicates that the stochastic exploration process is still present before the feasible connection region is reached. Therefore, the low variance in some final path-quality metrics does not mean that the algorithm is deterministic; rather, it reflects the stabilizing effect of the path post-processing mechanism under this specific map configuration.

The proposed method did not always achieve the shortest path or the shortest planning time. In the narrow-passage scenario, it produced a longer path than the other baselines because the safety-boundary mechanism encouraged a more conservative path with larger

obstacle clearance. Nevertheless, the method achieved better geometric smoothness and much larger minimum obstacle clearance in this scenario. Overall, these results show that the proposed method provides balanced performance in terms of path length, planning time, tree compactness, obstacle clearance, and geometric smoothness.

Path-planning comparison in the simple sparse scenario, complex dense scenario, narrow-passage scenario and highly cluttered scenario is shown in Figures 6–9.

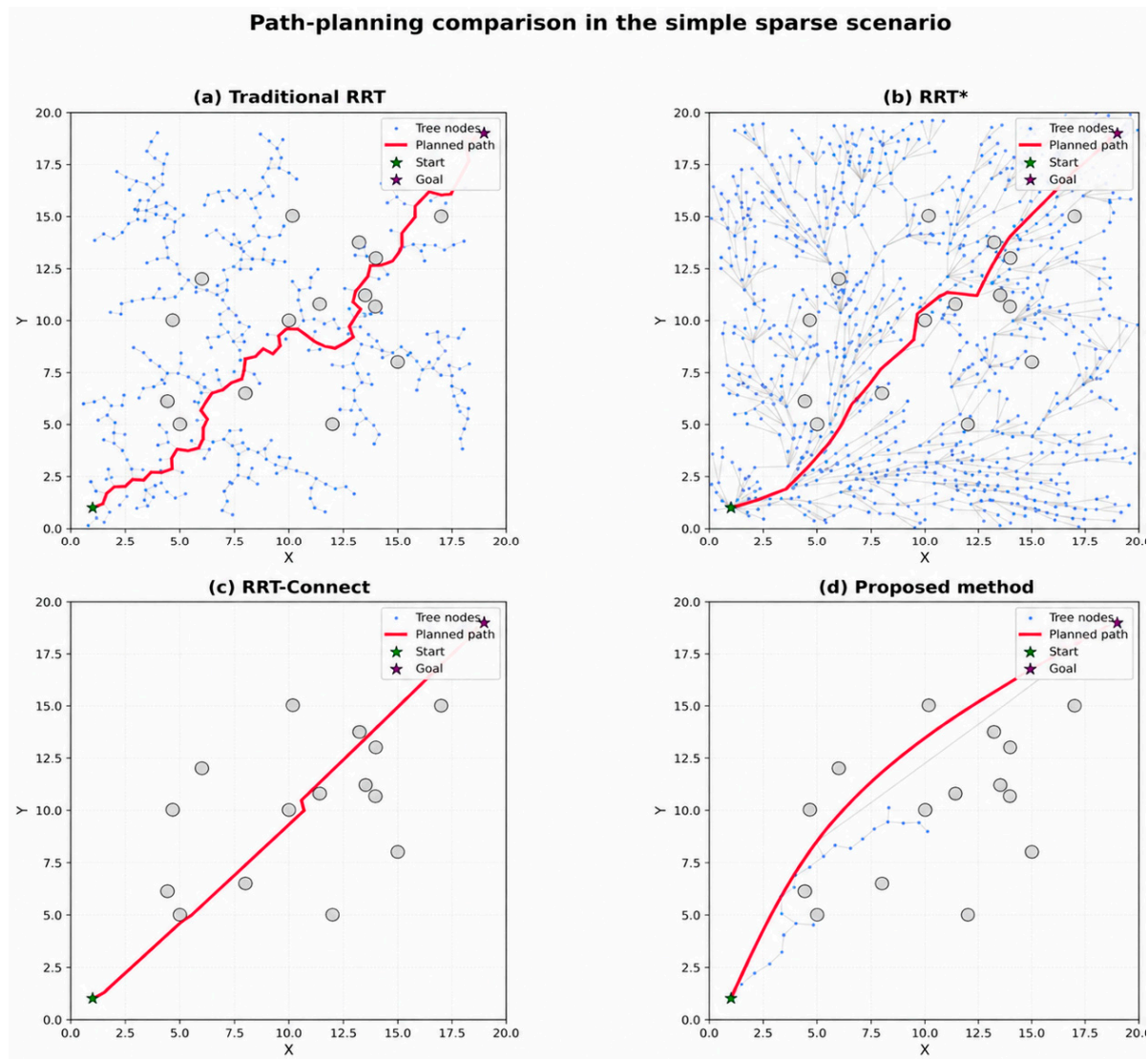


Figure 6. Path-planning comparison in the simple sparse scenario: (a) traditional RRT; (b) RRT*; (c) RRT-Connect; (d) proposed method. Gray circles indicate obstacles, gray lines indicate random tree branches, blue dots indicate tree nodes, red curves indicate planned paths, and the green and purple markers indicate the start and goal points, respectively.

4.5. Ablation Study

To evaluate the contribution of each module in the proposed framework, a module-wise ablation study was conducted. Six algorithm variants were designed: A0 Traditional RRT, A1 + KD-Tree, A2 + Adaptive Step, A3 + Safety Boundary, A4 + Pruning, and A5 Full Method. The ablation results are shown in Table 4.

The ablation results show that KD-Tree acceleration and adaptive step-size adjustment reduced redundant tree expansion and improved search efficiency. For example, in the complex dense scenario, the number of tree nodes decreased from 254.30 in A0 to 141.13 in A1 and 112.03 in A2. The safety-boundary module significantly improved obstacle

clearance. In the narrow-passage scenario, the minimum clearance increased from 0.0437 in A2 to 0.2601 in A3 after introducing the safety boundary.

The pruning module substantially shortened the path and increased obstacle clearance in several scenarios. For example, in the complex dense scenario, the path length decreased from 29.79 in A3 to 26.05 in A4. However, pruning alone may reduce geometric smoothness in narrow passages, as indicated by the decrease in mean turning angle in A4. After adding spline-based smoothing in A5, the mean turning angle in the narrow-passage scenario increased to 178.35° , and the turning-angle fluctuation was reduced. These results indicate that smoothing mainly improves geometric smoothness rather than strictly minimizing path length.

The full method achieved the clearest improvement in tree compactness, obstacle clearance, and geometric smoothness. Nevertheless, in the narrow-passage scenario, the full method produced a slightly longer path and required more planning time than some variants because the safety-boundary constraint led to a more conservative path. This result is consistent with the design purpose of the proposed framework, which aims to balance path safety, smoothness, and computational efficiency rather than optimize a single metric.

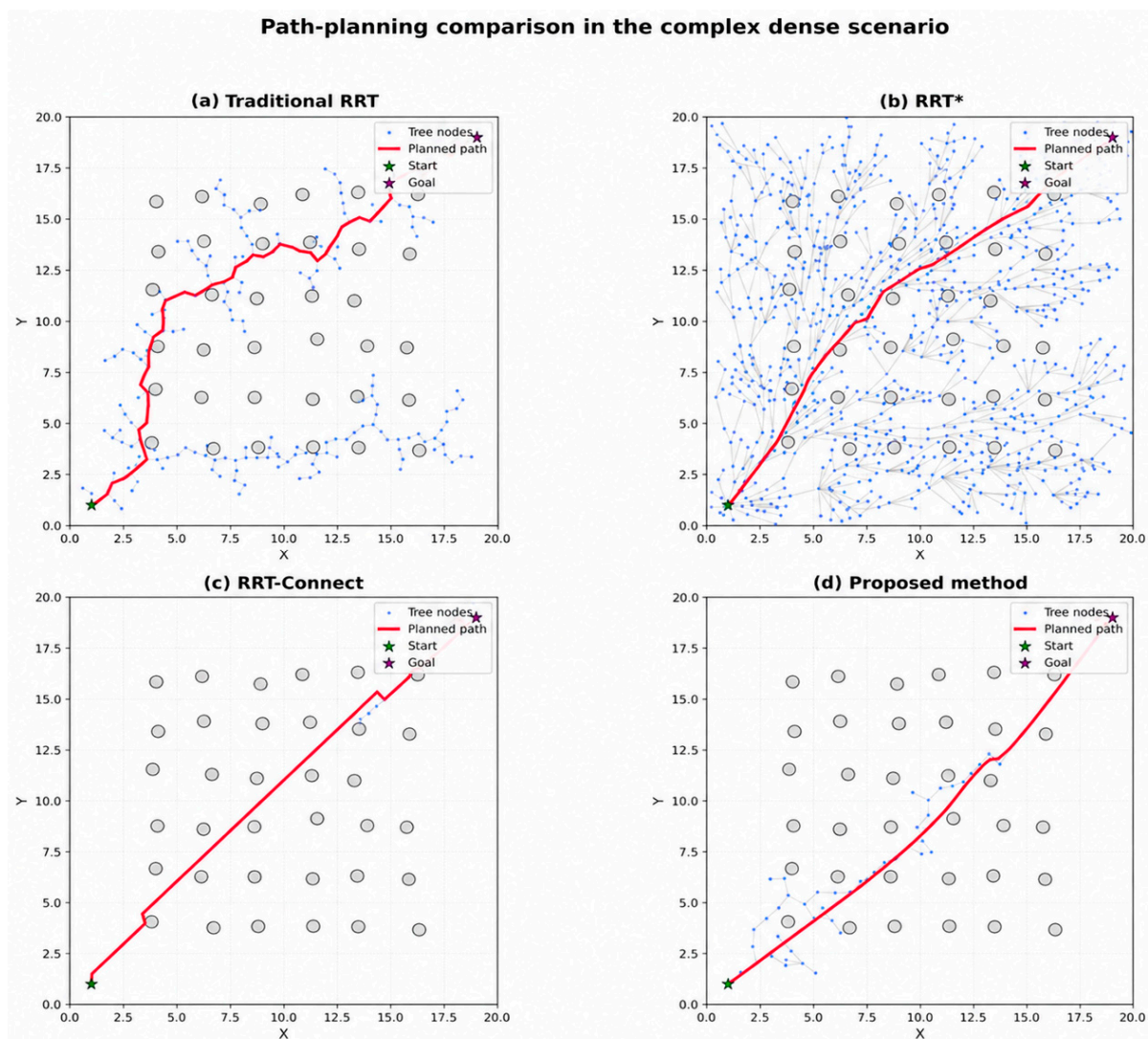


Figure 7. Path-planning comparison in the complex dense scenario: (a) traditional RRT; (b) RRT*; (c) RRT-Connect; (d) proposed method. Gray circles indicate obstacles, gray lines indicate random tree branches, blue dots indicate tree nodes, red curves indicate planned paths, and the green and purple markers indicate the start and goal points, respectively.

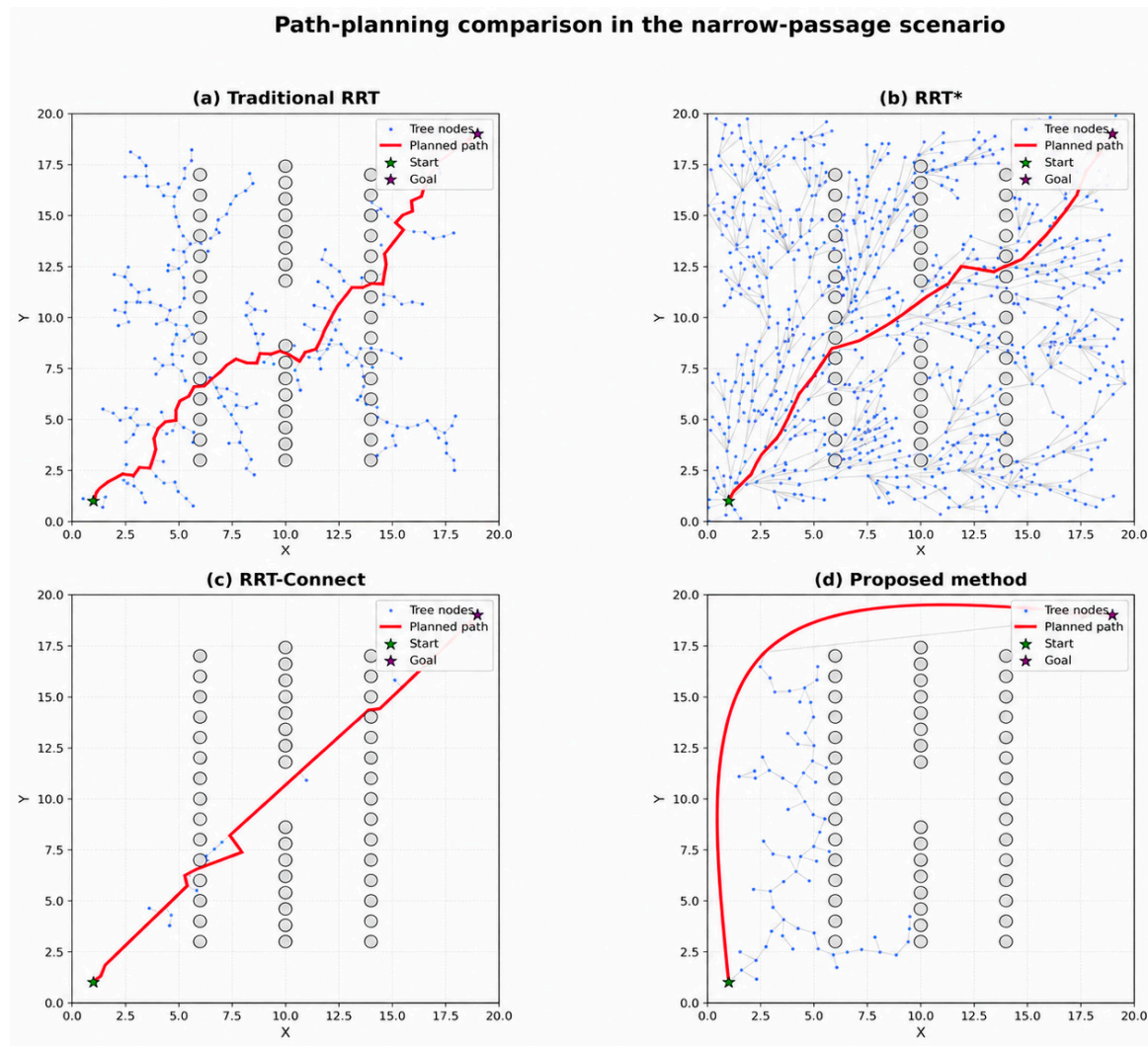


Figure 8. Path-planning comparison in the narrow-passage scenario: (a) traditional RRT; (b) RRT*; (c) RRT-Connect; (d) proposed method. Gray circles indicate obstacles, gray lines indicate random tree branches, blue dots indicate tree nodes, red curves indicate planned paths, and the green and purple markers indicate the start and goal points, respectively.

Table 4. Ablation study of different modules in the proposed framework. Values are reported as mean $\pm$ standard deviation over 30 independent runs.

Scenario	Variant	Success Rate (%)	Path Length	Time (s)	Tree Nodes	Min Clearance	Mean Angle (°)
Complex dense	A0 Traditional RRT	100	30.58 $\pm$ 1.20	0.051 $\pm$ 0.019	254.30 $\pm$ 53.69	0.078 $\pm$ 0.056	148.57 $\pm$ 3.33
Complex dense	A1 + KD-Tree	100	29.14 $\pm$ 1.29	0.021 $\pm$ 0.009	141.13 $\pm$ 39.52	0.088 $\pm$ 0.065	154.37 $\pm$ 3.30
Complex dense	A2 + Adaptive Step	100	28.90 $\pm$ 0.77	0.020 $\pm$ 0.004	112.03 $\pm$ 22.02	0.070 $\pm$ 0.064	154.64 $\pm$ 4.20
Complex dense	A3 + Safety Boundary	100	29.79 $\pm$ 1.42	0.028 $\pm$ 0.007	116.07 $\pm$ 26.86	0.249 $\pm$ 0.042	152.77 $\pm$ 4.51
Complex dense	A4 + Pruning	100	26.05 $\pm$ 0.54	0.113 $\pm$ 0.031	118.57 $\pm$ 29.61	0.529 $\pm$ 0.214	157.13 $\pm$ 12.16
Complex dense	A5 Full Method	100	26.88 $\pm$ 1.17	0.107 $\pm$ 0.035	41.83 $\pm$ 19.27	0.427 $\pm$ 0.314	155.86 $\pm$ 29.83
Narrow passage	A0 Traditional RRT	100	31.41 $\pm$ 1.67	0.070 $\pm$ 0.028	290.13 $\pm$ 69.74	0.054 $\pm$ 0.033	146.75 $\pm$ 3.35
Narrow passage	A1 + KD-Tree	100	29.39 $\pm$ 1.16	0.023 $\pm$ 0.005	146.50 $\pm$ 20.49	0.044 $\pm$ 0.036	153.10 $\pm$ 3.79
Narrow passage	A2 + Adaptive Step	100	29.92 $\pm$ 1.53	0.027 $\pm$ 0.012	134.80 $\pm$ 45.10	0.044 $\pm$ 0.028	152.32 $\pm$ 4.56

Table 4. Cont.

Scenario	Variant	Success Rate (%)	Path Length	Time (s)	Tree Nodes	Min Clearance	Mean Angle (°)
Narrow passage	A3 + Safety Boundary	100	37.15 ± 1.55	0.078 ± 0.041	245.83 ± 117.17	0.260 ± 0.081	150.37 ± 3.95
Narrow passage	A4 + Pruning	100	31.43 ± 1.03	0.161 ± 0.060	257.77 ± 137.39	0.989 ± 0.595	115.33 ± 16.38
Narrow passage	A5 Full Method	100	32.50 ± 0.86	0.121 ± 0.018	87.70 ± 23.15	1.048 ± 0.449	178.35 ± 0.06
Highly cluttered	A0 Traditional RRT	100	30.57 ± 1.38	0.067 ± 0.022	262.63 ± 57.57	0.065 ± 0.076	148.13 ± 3.98
Highly cluttered	A1 + KD-Tree	100	29.31 ± 1.04	0.023 ± 0.006	135.77 ± 27.33	0.080 ± 0.124	153.55 ± 4.13
Highly cluttered	A2 + Adaptive Step	100	29.10 ± 1.11	0.023 ± 0.006	105.00 ± 24.74	0.083 ± 0.064	154.46 ± 5.02
Highly cluttered	A3 + Safety Boundary	100	29.72 ± 1.30	0.036 ± 0.013	115.23 ± 34.46	0.245 ± 0.045	153.29 ± 3.82
Highly cluttered	A4 + Pruning	100	25.46 ± 0.00	0.051 ± 0.034	148.77 ± 94.33	3.095 ± 0.727	180.00 ± 0.00
Highly cluttered	A5 Full Method	100	25.46 ± 0.00	0.022 ± 0.005	4.43 ± 3.39	3.227 ± 0.779	180.00 ± 0.00

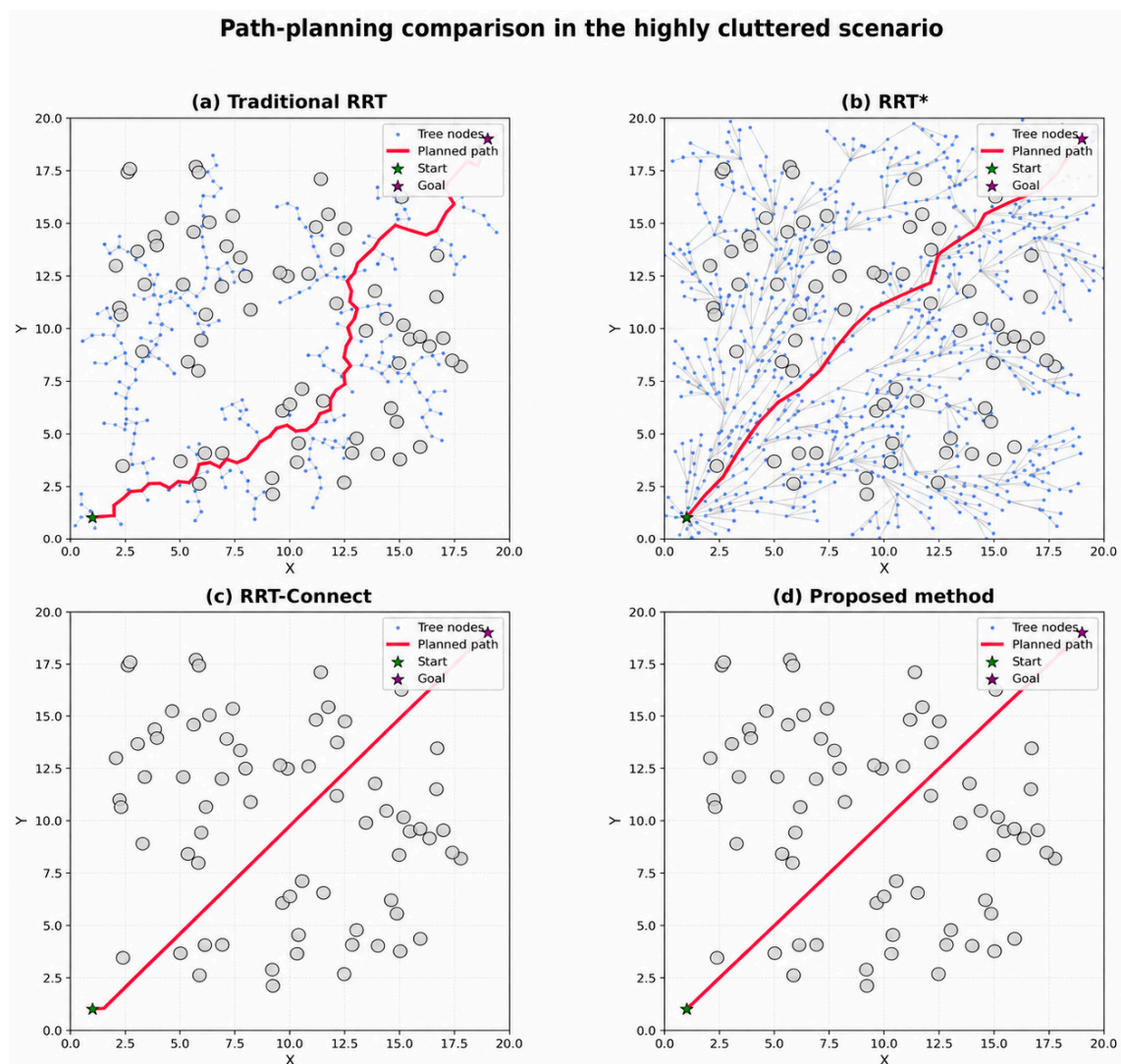


Figure 9. Path-planning comparison in the highly cluttered scenario: (a) traditional RRT; (b) RRT*; (c) RRT-Connect; (d) proposed method. Gray circles indicate obstacles, gray lines indicate random tree branches, blue dots indicate tree nodes, red curves indicate planned paths, and the green and purple markers indicate the start and goal points, respectively.

4.6. Parameter Sensitivity Analysis

A parameter sensitivity analysis was conducted to justify the selection of key parameters, including the balancing factor α , the safety margin, and the spline sampling number. The sensitivity experiments were performed in the complex dense and narrow-passage scenarios, and each parameter setting was independently tested 30 times. The results are shown in Tables 5–7.

Table 5. Sensitivity analysis of the balancing factor α . Values are reported as mean $\pm$ standard deviation over 30 independent runs.

Scenario	Value	Path Length	Time (s)	Tree Nodes	Min Clearance	Mean Angle (°)
Complex dense	0.40	26.08 $\pm$ 0.78	0.106 $\pm$ 0.035	36.17 $\pm$ 20.61	0.422 $\pm$ 0.218	162.21 $\pm$ 21.55
Complex dense	0.50	26.45 $\pm$ 1.12	0.114 $\pm$ 0.031	43.67 $\pm$ 18.76	0.446 $\pm$ 0.355	162.12 $\pm$ 20.78
Complex dense	0.60	26.89 $\pm$ 1.82	0.104 $\pm$ 0.036	40.73 $\pm$ 28.16	0.467 $\pm$ 0.291	158.19 $\pm$ 24.98
Complex dense	0.70	26.45 $\pm$ 1.23	0.103 $\pm$ 0.033	37.97 $\pm$ 22.74	0.408 $\pm$ 0.223	163.89 $\pm$ 21.82
Complex dense	0.80	26.18 $\pm$ 0.76	0.121 $\pm$ 0.042	39.97 $\pm$ 18.85	0.377 $\pm$ 0.155	168.01 $\pm$ 17.81
Narrow passage	0.40	32.94 $\pm$ 0.89	0.121 $\pm$ 0.015	87.30 $\pm$ 21.80	1.234 $\pm$ 0.451	178.33 $\pm$ 0.07
Narrow passage	0.50	33.04 $\pm$ 0.83	0.118 $\pm$ 0.015	80.63 $\pm$ 16.31	1.307 $\pm$ 0.433	178.32 $\pm$ 0.07
Narrow passage	0.60	32.72 $\pm$ 1.05	0.126 $\pm$ 0.028	90.57 $\pm$ 31.92	1.126 $\pm$ 0.528	176.64 $\pm$ 8.99
Narrow passage	0.70	32.80 $\pm$ 0.94	0.125 $\pm$ 0.024	90.43 $\pm$ 27.44	1.256 $\pm$ 0.543	178.33 $\pm$ 0.08
Narrow passage	0.80	32.68 $\pm$ 1.03	0.122 $\pm$ 0.017	87.13 $\pm$ 23.01	1.183 $\pm$ 0.509	176.72 $\pm$ 8.82

Table 6. Sensitivity analysis of the safety margin. Values are reported as mean $\pm$ standard deviation over 30 independent runs.

Scenario	Value	Path Length	Time (s)	Tree Nodes	Min Clearance	Mean Angle (°)
Complex dense	0.10	26.02 $\pm$ 0.70	0.089 $\pm$ 0.028	26.30 $\pm$ 12.68	0.447 $\pm$ 0.479	165.58 $\pm$ 23.99
Complex dense	0.15	25.93 $\pm$ 0.65	0.102 $\pm$ 0.036	30.97 $\pm$ 16.66	0.331 $\pm$ 0.128	171.42 $\pm$ 18.45
Complex dense	0.20	26.70 $\pm$ 1.40	0.103 $\pm$ 0.029	35.40 $\pm$ 17.30	0.463 $\pm$ 0.289	157.79 $\pm$ 32.80
Complex dense	0.25	27.02 $\pm$ 1.52	0.116 $\pm$ 0.028	47.20 $\pm$ 19.37	0.524 $\pm$ 0.273	154.95 $\pm$ 25.22
Complex dense	0.30	26.97 $\pm$ 1.07	0.125 $\pm$ 0.029	55.47 $\pm$ 21.74	0.458 $\pm$ 0.114	148.20 $\pm$ 25.58
Narrow passage	0.10	30.63 $\pm$ 3.51	0.113 $\pm$ 0.018	84.60 $\pm$ 26.53	0.374 $\pm$ 0.447	166.07 $\pm$ 18.84
Narrow passage	0.15	30.93 $\pm$ 2.57	0.131 $\pm$ 0.027	103.13 $\pm$ 30.22	0.405 $\pm$ 0.389	167.17 $\pm$ 19.06
Narrow passage	0.20	32.72 $\pm$ 1.29	0.123 $\pm$ 0.018	88.13 $\pm$ 22.74	1.201 $\pm$ 0.520	176.58 $\pm$ 9.60
Narrow passage	0.25	33.08 $\pm$ 0.85	0.121 $\pm$ 0.014	77.67 $\pm$ 15.72	1.342 $\pm$ 0.418	178.34 $\pm$ 0.07
Narrow passage	0.30	32.64 $\pm$ 0.82	0.133 $\pm$ 0.021	90.07 $\pm$ 24.09	1.190 $\pm$ 0.467	178.33 $\pm$ 0.06

The sensitivity results indicate that the optimal value of α varies across scenarios. Therefore, $\alpha = 0.6$ is selected as a balanced configuration rather than a universally optimal value. This value provides a practical balance between path length, obstacle clearance, and tree scale in the tested scenarios.

A smaller safety margin tends to generate shorter paths but lower obstacle clearance, while a larger safety margin improves safety at the cost of longer and more conservative paths. Therefore, a safety margin of 0.20 was selected as a compromise between path length and obstacle clearance.

Increasing the number of spline samples improves geometric smoothness. Compared with 30 samples, 60 samples substantially reduce turning-angle fluctuation in the narrow-

passage scenario, while the improvement from 60 to 120 samples becomes relatively limited. Therefore, 60 samples were adopted to balance smoothness and computational cost.

Table 7. Sensitivity analysis of the spline sampling number. Values are reported as mean $\pm$ standard deviation over 30 independent runs.

Scenario	Value	Path Length	Time (s)	Tree Nodes	Min Clearance	Mean Angle (°)
Complex dense	20	26.21 $\pm$ 0.79	0.105 $\pm$ 0.034	37.93 $\pm$ 16.20	0.428 $\pm$ 0.301	163.85 $\pm$ 20.88
Complex dense	30	26.17 $\pm$ 0.68	0.118 $\pm$ 0.041	38.33 $\pm$ 20.84	0.373 $\pm$ 0.179	162.78 $\pm$ 23.32
Complex dense	60	26.23 $\pm$ 0.82	0.114 $\pm$ 0.032	42.83 $\pm$ 21.68	0.449 $\pm$ 0.286	160.72 $\pm$ 23.07
Complex dense	120	26.53 $\pm$ 0.93	0.112 $\pm$ 0.037	43.87 $\pm$ 24.06	0.419 $\pm$ 0.324	163.64 $\pm$ 22.72
Narrow passage	20	32.48 $\pm$ 0.89	0.122 $\pm$ 0.016	89.27 $\pm$ 23.31	1.115 $\pm$ 0.484	174.10 $\pm$ 3.24
Narrow passage	30	32.96 $\pm$ 1.18	0.124 $\pm$ 0.019	89.67 $\pm$ 25.13	1.189 $\pm$ 0.456	175.15 $\pm$ 7.89
Narrow passage	60	32.85 $\pm$ 1.10	0.123 $\pm$ 0.015	87.40 $\pm$ 18.18	1.102 $\pm$ 0.584	178.35 $\pm$ 0.08
Narrow passage	120	32.71 $\pm$ 1.04	0.133 $\pm$ 0.015	89.07 $\pm$ 20.52	1.050 $\pm$ 0.489	179.14 $\pm$ 0.28

4.7. Limitations and Discussion

Although the proposed method improves tree compactness, obstacle clearance, and geometric smoothness, several limitations remain. First, the current experiments are conducted in two-dimensional static environments with circular obstacles, and dynamic obstacles are not considered. Second, the proposed planner is a geometric path-planning method and does not explicitly model nonholonomic constraints, steering limits, curvature bounds, or vehicle dynamics. Therefore, the generated paths should be regarded as smooth geometric reference paths rather than dynamically feasible trajectories.

Third, in extremely narrow or highly constrained environments, the safety-boundary mechanism may generate more conservative paths, resulting in longer path length and increased planning time. This phenomenon is observed in the narrow-passage scenario, where the method prioritizes larger obstacle clearance and geometric smoothness over strict path-length minimization. Finally, the current experiments are conducted in low-dimensional environments, and further validation is still required for high-dimensional robotic systems and real-world applications. Future work will extend the proposed framework to dynamic environments, high-dimensional configuration spaces, and robot-specific kinematic models.

5. Conclusions

This paper proposed an integrated multi-strategy improved RRT framework for robot path planning. The framework combines KD-Tree acceleration, adaptive step-size adjustment, safety-boundary collision checking, direct goal-connection, safety-validated shortcut pruning, and spline-based path smoothing. To evaluate its effectiveness, comparative experiments were conducted against traditional RRT, RRT*, and RRT-Connect in four representative scenarios, including simple sparse, complex dense, narrow-passage, and highly cluttered environments. Each algorithm was independently tested 30 times, and the results were reported using mean and standard deviation.

The experimental results show that all algorithms achieved a 100% success rate in the tested scenarios. RRT* generated near-optimal paths but required substantially longer planning time and a much larger search tree, while RRT-Connect achieved the shortest planning time but provided relatively small obstacle clearance. In contrast, the proposed method achieved a better balance among path length, tree compactness, obstacle clearance, geometric smoothness, and computational efficiency. Compared with traditional RRT,

it substantially reduced redundant tree expansion in most scenarios and achieved the largest minimum obstacle clearance among the tested algorithms. The turning-angle-based smoothness metrics also indicate that the resulting paths contain fewer sharp turns and provide smoother geometric reference paths.

The ablation study further verified the contribution of each module in the proposed framework. KD-Tree acceleration and adaptive step-size adjustment help reduce redundant exploration, safety-boundary checking improves obstacle clearance, and pruning and spline smoothing contribute to path compactness and geometric smoothness. The parameter sensitivity analysis shows that the balancing factor, safety margin, and spline sampling number affect the trade-off among path length, safety, smoothness, and computational cost. Therefore, the selected parameters are not claimed to be universally optimal, but are adopted as balanced settings under the tested scenarios.

Despite these improvements, several limitations remain. The current experiments were conducted in two-dimensional static environments with circular obstacles, and dynamic obstacles were not considered. In addition, the planner is a geometric path-planning method and does not explicitly model nonholonomic constraints, steering limits, curvature bounds, or vehicle dynamics. Therefore, the generated paths should be regarded as smooth geometric reference paths rather than dynamically feasible trajectories. Future work will extend the proposed framework to dynamic environments, high-dimensional configuration spaces, and robot-specific kinematic models, and will further validate its performance in real robotic systems.

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

Funding: This research received no external funding. The APC was funded by the authors.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data supporting the findings of this study are contained within the article. The simulation data and code generated and analyzed during the current study are available from the corresponding author upon reasonable request.

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

References

1. Alzahrani, A.; Almutairi, S. Path planning for mobile robots in dynamic environments using enhanced RRT algorithm. *Int. J. Adv. Comput. Sci. Appl.* **2022**, *13*, 689–701. [\[CrossRef\]](#)
2. Elhoseny, M.; Abdelaziz, A.; Sallam, A. Optimal path planning for autonomous vehicles using RRT-based algorithms. *Int. J. Adv. Comput. Sci. Appl.* **2021**, *12*, 543–552.
3. Zhuang, M.; Li, G.; Ding, K. Obstacle avoidance path planning for apple picking robotic arm incorporating artificial potential field and A* algorithm. *IEEE Access* **2023**, *11*, 100070–100082. [\[CrossRef\]](#)
4. Wei, K.; Ren, B. A method on dynamic path planning for robotic manipulator autonomous obstacle avoidance based on an improved RRT algorithm. *Sensors* **2018**, *18*, 571. [\[CrossRef\]](#) [\[PubMed\]](#)
5. Kang, M.; Chen, Q.; Fan, Z.; Yu, C.; Wang, Y.; Yu, X. A RRT based path planning scheme for multi-DOF robots in unstructured environments. *Comput. Electron. Agric.* **2024**, *218*, 108707. [\[CrossRef\]](#)
6. Gan, X.; Huo, Z.; Li, W. DP-A*: For path planning of UGV and contactless delivery. *IEEE Trans. Intell. Transp. Syst.* **2024**, *25*, 907–919. [\[CrossRef\]](#)
7. Khatib, O. Real-time obstacle avoidance for manipulators and mobile robots. *Int. J. Robot. Res.* **1986**, *5*, 90–98. [\[CrossRef\]](#)

8. Alotaibi, F.; Alshehri, S. Path smoothing of RRT algorithm using B-spline curve for mobile robots. *Int. J. Adv. Comput. Sci. Appl.* **2022**, *13*, 456–465.
9. Lin, K.; Li, Y.; Chen, S.; Li, D.; Wu, X. Motion planner with fixed-horizon constrained reinforcement learning for complex autonomous driving scenarios. *IEEE Trans. Intell. Veh.* **2024**, *9*, 1577–1588. [[CrossRef](#)]
10. LaValle, S.M.; Kuffner, J.J. Rapidly-exploring random trees: Progress and prospects. In *Algorithmic and Computational Robotics: New Directions*; Donald, B.R., Lynch, K.M., Rus, D., Eds.; A K Peters: Wellesley, MA, USA, 2001; pp. 293–308.
11. LaValle, S.M. *Rapidly-Exploring Random Trees: A New Tool for Path Planning*; Technical Report TR 98-11; Iowa State University: Ames, IA, USA, 1998.
12. Wang, Y.; Jiang, W.; Luo, Z.; Yang, L.; Wang, Y. Path planning of a 6-DOF measuring robot with a direction guidance RRT method. *Expert Syst. Appl.* **2024**, *238*, 122057. [[CrossRef](#)]
13. Alharbi, S.; Alshammari, Y. Fast nearest neighbor search for RRT algorithm using KD-Tree. *Int. J. Adv. Comput. Sci. Appl.* **2022**, *13*, 289–298.
14. Wu, Z.P.; Meng, Z.J.; Zhao, W.L.; Wu, Z. Fast-RRT: A RRT-based optimal path finding method. *Appl. Sci.* **2021**, *11*, 11777. [[CrossRef](#)]
15. Kuffner, J.J.; LaValle, S.M. RRT-Connect: An efficient approach to single-query path planning. In Proceedings of the 2000 IEEE International Conference on Robotics and Automation, San Francisco, CA, USA, 24–28 April 2000; pp. 995–1001.
16. Karaman, S.; Frazzoli, E. Sampling-based algorithms for optimal motion planning. *Int. J. Robot. Res.* **2011**, *30*, 846–894. [[CrossRef](#)]
17. Islam, F.; Nasir, J.; Malik, U.; Ayaz, Y.; Hasan, O. RRT*-Smart: Rapid convergence implementation of RRT* towards optimal solution. In Proceedings of the 2012 IEEE International Conference on Mechatronics and Automation, Chengdu, China, 5–8 August 2012; pp. 1651–1656.
18. Cui, X.; Cui, X.; Wang, C.; 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. [[CrossRef](#)]
19. Douglas, D.H.; Peucker, T.K. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica* **1973**, *10*, 112–122. [[CrossRef](#)]
20. Dolgov, D.; Thrun, S.; Montemerlo, M.; Diebel, J. Path planning for autonomous vehicles in unknown semi-structured environments. *Int. J. Robot. Res.* **2010**, *29*, 485–501. [[CrossRef](#)]
21. Cosenza, C.; Malfi, P.; Nicoletta, A.; Niola, V.; Savino, S. Multibody modeling for the analysis of the maneuverability of a skid steering 6-wheeled robot. In *Advances in Italian Mechanism Science*; Quaglia, G., Boschetti, G., Carbone, G., Eds.; Mechanisms and Machine Science; Springer: Cham, Switzerland, 2024; Volume 164. [[CrossRef](#)]
22. Li, X.; Tong, Y. Path planning of a mobile robot based on the improved RRT algorithm. *Appl. Sci.* **2024**, *14*, 25. [[CrossRef](#)]
23. Oh, T.; Won, Y.-J.; Lee, S. TA-RRT*: Adaptive sampling-based path planning using terrain analysis. *Appl. Sci.* **2025**, *15*, 2287. [[CrossRef](#)]
24. Faroni, M.; Pedrocchi, N.; Beschi, M. Adaptive hybrid local-global sampling for fast informed sampling-based optimal path planning. *Auton. Robot.* **2024**, *48*, 6. [[CrossRef](#)]
25. 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. [[CrossRef](#)]
26. Fan, H.; Huang, J.; Huang, X.; Zhu, H.; Su, H. BI-RRT*: An improved path planning algorithm for secure and trustworthy mobile robots systems. *Heliyon* **2024**, *10*, e26403. [[CrossRef](#)] [[PubMed](#)]
27. Lei, S.; Li, T.; Gao, X.; Xue, P.; Song, G. Research on improved RRT path planning algorithm based on multi-strategy fusion. *Sci. Rep.* **2025**, *15*, 13312. [[CrossRef](#)] [[PubMed](#)]

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.