Research on Path Planning of Automated Parking Robots Based on Improved RRT Algorithm

Zichuan Wang[1], Yawen Fan[2,*], Jingfeng Shen[1,*], Shikun Zhang[1], Fangting Liu[1]
[1]*School of Mechanical Engineering, University of Shanghai for Science and Technology, Shanghai, China*
[2]*Sino-British International College, University of Shanghai for Science and Technology, Shanghai, China*
**Corresponding Author*

Abstract: This paper proposed an improved RRT path planning algorithm based on child reconnection (RRT with Child Reconnection, RRT-CRe). It addresses the limitations of the basic Rapidly Expanding Random Tree (RRT) algorithm in narrow, compact environments like smart garages, where it suffers from blind expansion and low sampling point utilization during parking path planning. A realistic three-dimensional environment model was first formulated for a selected intelligent garage zone under weekday peak-hour conditions, enabling accurate representation of real-time occupancy. Building upon this environment, a goal-biased sampling and expansion strategy enhanced by a child-reconnection mechanism was proposed to overcome the inefficiency of conventional RRT. To further ensure feasibility under practical constraints, a circumscribed-circle model was integrated into the local rewiring phase, capturing the geometric limitations of automated parking robots and their surroundings. Results show that in narrow and complex environments involving multiple target points, the proposed RRT-CRe algorithm substantially outperforms the basic RRT, achieving an average reduction of approximately 55% in parking time and an improvement of about 45% in success rate. Moreover, when compared with RRT*, a widely used derivative of the basic RRT, RRT-CRe still exhibits an increase of approximately 35% in success rate. These results collectively demonstrate that the proposed algorithm satisfies the efficiency and reliability requirements of automated parking robots in challenging scenarios.

Keywords: RRT; Path Planning; Intelligent Garage; Automated Parking Robot

1. Introduction

In recent years, with the continuous improvement of living standards among Chinese residents, the number of vehicles in circulation has steadily increased. However, increasingly scarce land resources coupled with outdated parking space construction standards have resulted in a severe shortage of parking spaces in most cities [1]. Against the backdrop of the accelerated integration of smart cities and intelligent transportation, a variety of parking facilities and technical solutions have emerged, including traditional underground parking garages, vertical circulation stereo parking spaces, aisle stacking parking systems, and intelligent parking garages based on automatic parking robots.

Among these solutions, automatic parking robots are highly aligned with the national strategic expectations for smart city construction. According to public data released by multiple enterprises in the market, the application of Automated Guided Vehicles (AGVs) can increase parking lot utilization by at least 40%, demonstrating broad application prospects [2]. In terms of space utilization, intelligent parking garages based on automatic parking robots eliminate pedestrian walkways; reduce inter-vehicle gaps, and lower floor-to-floor heights. Consequently, the number of parking spaces that can be constructed under the same floor area is significantly increased, while the corresponding construction costs are reduced accordingly.

In traditional parking garages, drivers are required to search for parking spaces manually without effective information guidance mechanisms, resulting in low parking efficiency. In contrast, intelligent parking garages based on automatic parking robots transform the

"driver-searches-for-space" model into a "space-waits-for-driver" model through system-level scheduling. Additionally, they can synchronize real-time parking space availability information, enabling transparent parking status visibility for both users and managers. This significantly enhances parking convenience and management efficiency [3]. Therefore, as shown in Figure 1, the development of smart parking facilities based on automated parking robots has become one of the key technological approaches to alleviating the urban parking supply-demand imbalance [4]; and the efficiency with which the automatic parking robot (shown in Fig. 2) can complete parking tasks largely depends on the performance of its path planning [5].

Figure 1. Intelligent Garage Scenario Based on Automatic Parking Robots

1-Unloaded Automatic Parking Robot
2-Vehicle-Carrying Automatic Parking Robot

Figure 2. Automatic Parking Robot and Target Vehicle to Be Parked

1-Unloaded Automatic Parking Robot
2-Vehicle-Carrying Automatic Parking Robot

Currently, path planning for automated parking robots primarily relies on graph search algorithms and sampling-based algorithms. Graph search algorithms, such as A*, Dijkstra, can ensure path optimality in 2D parking garage scenarios. However, when applied to multidimensional spatial environments featuring multi-story buildings and narrow alleys, they are prone to node explosions, leading to a sharp decline in real-time performance [6]. In contrast, sampling-based algorithms, such as the basic Rapidly-exploring Random Tree (RRT) algorithm, rely on probabilistic exploration mechanisms to maintain efficient scaling capabilities in high-dimensional environments [7], making them more suitable for parking scenarios in intelligent parking garages.

Therefore, numerous researchers have adopted the RRT algorithm and developed various optimized variants based on its inherent advantages. The Quick RRT* (Q-RRT*) algorithm minimizes path cost by leveraging the triangle inequality property, which involves considering both neighboring points and their ancestor nodes during parent node reselection and path reconnection. However, the depth values of ancestral nodes in this algorithm are arbitrarily set, resulting in poor adaptability across different scenarios [8]. The Stack-RRT* algorithm generates nodes closer to obstacles through a binary classification method, thereby expanding the search range of potential parent nodes, reducing initial path costs, and enhancing the efficiency of random tree expansion. However, generating paths for nodes near obstacles relies heavily on collision detection, which incurs additional computational time [9].

Other studies have combined Random Tree Search (RRT) with Artificial Potential Field (APF) methods. Considering the inherent randomness of tree growth in the RRT* algorithm, an approach that integrates APF-guided sampling with RRT* is proposed to enhance the directionality of the sampling process [10]. The APF algorithm features a simple structure and excellent real-time performance, but path oscillation may occur in areas with dense obstacles [11,12]. To enhance robots' path planning capabilities in complex environments, the AMP-RRT* algorithm integrates heuristic backtracking with a local obstacle avoidance mechanism, thereby significantly improving path search efficiency and global planning performance in complex static environments [13]. Additionally, hybrid methods have garnered significant attention in recent years. For instance, a hybrid Adapted-RRT (Adaptive Randomized Rapid Tracking) approach is proposed that combines sampling with meta-heuristic algorithms to address three-dimensional path planning problems [14]. However, this approach typically relies on massive amounts of training data and is sensitive to reward design, making it less suitable for real-world parking tasks.

Given the significant advantages of RRT series algorithms in computational cost and high-dimensional adaptability, this study adopts an improved RRT algorithm for the path

planning of automatic parking robots. Corresponding optimization strategies are proposed to address the issues of blind expansion and low sampling point utilization inherent in the basic RRT algorithm.

This paper presents a RRT with Child Reconnectionalgorithm (RRT-CRe). The core of this algorithm lies in introducing a goal-biased strategy optimized by a sub-reconstruction mechanism into the sampling and expansion phases of the basic RRT algorithm. This design achieves path quality optimization while maintaining the algorithm's real-time performance, thereby enhancing its adaptability to various complex scenarios. Meanwhile, the obstacle avoidance phase of the algorithm is optimized according to the physical dimensions of the automatic parking robot and its operating environment, making it more suitable for lightweight and efficient intelligent parking garage systems.

To verify the effectiveness of the RRT-CRe algorithm, a 3D intelligent parking garage environment is simulated in MATLAB. Performance differences among the proposed algorithm, the basic RRT algorithm, and the RRT* algorithm are evaluated using three key metrics: average path length, average parking time, and planning success rate.

2. Problem Initialization and Environment Modeling

2.1 Problem Initialization

Traditional RRT algorithms do not account for robot dimensions during scaling, making automated parking robots prone to colliding with obstacles while navigating algorithmically planned paths. As shown in Figure 3, an external circle model for the automated parking robot was established to address this issue. This incorporates the dimensions of both the vehicle body and the automated parking robot into the algorithm, thereby improving practical application. The expression for diameter D is given by Equation (1):

$$D = \sqrt{L^2 + W^2} \quad (1)$$

To simulate the three-dimensional environment of an intelligent garage, the following initialization steps are performed: 1) The three-dimensional layout of the garage environment must align with real-world garage settings, with reasonable spacing between parking bays and basic symmetry maintained; 2) While ensuring sufficient width for the automated parking robot to avoid collisions and accommodating its maximum turning radius, minimize the width of the driving lanes as much as possible; 3) Assume the automated parking robot travels at a constant speed, and disregard its dwell time when calculating the average path planning duration; 4) Model the automated parking robot as a rectangular prism, with the point-based robot in MATLAB representing the center point of this prism; 5) In this 3D environment, cylinders simulate pillar obstacles within the garage, while cubes and rectangular prisms represent different types of vehicles parked in the garage.

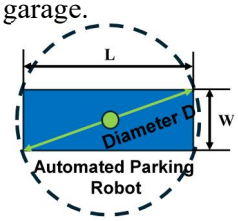
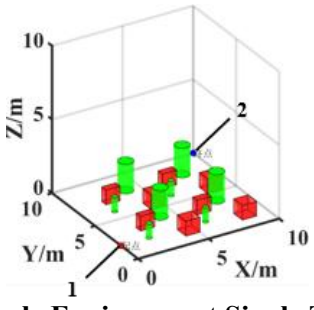


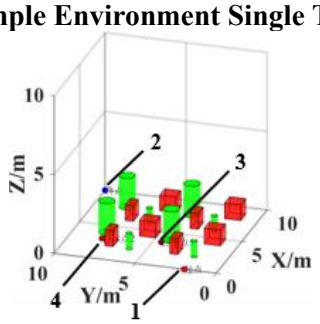
Figure 3. External Circular Model for Automatic Parking Robot

2.2 Environment Modeling

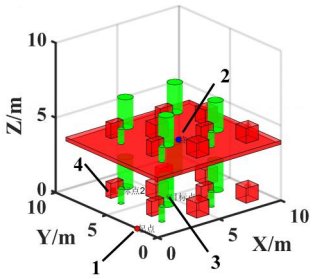
As shown in Figure 4, three distinct 3D planning environments were constructed: a simple environment with a single target point, a simple environment with multiple target points, and a narrow complex environment with multiple target points. Among these, the narrow complex environment with multiple target points most closely resembles the intelligent garage environment in which automated parking robots operate.



a) Simple Environment Single Target



b) Simple Environment Multi-Target Points



c) Multiple Target Points in a Narrow and Complex Environment

Figure 4. Three Different 3D Planning Environments

1-Starting Point
2-Ending Point
3-Target Point 1
4-Target Point 2

All maps measure 10m × 10m × 10m. In simple environments, the start and end points are positioned at opposite ends of the map: the start point coordinates are (0, 2, 0) m, and the end point coordinates are (10, 10, 0) m. In narrow, complex environments, the end point is placed at the center of the second floor. The algorithm's maximum iteration count is set to 5000, with other variable parameters adjusted accordingly for different maps.

3. Basic RRT Algorithm

The Rapidly-exploring Random Tree (RRT) algorithm is a path planning method based on sampling. It iteratively generates randomly sampled points in free space and greedily extends the nearest node toward the sampling direction by a fixed step size, thereby rapidly constructing a search tree that covers the feasible region. The algorithm incorporates new nodes into the tree structure only after collision detection confirms the extension path. Search terminates when a new node enters the target neighborhood, and the initial path is reconstructed by backtracking along parent pointers. Figure 5 illustrates the basic RRT algorithm extension.

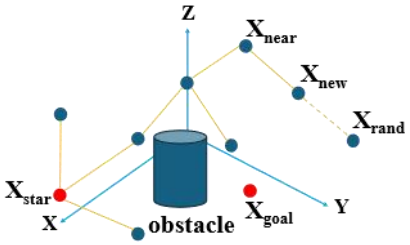


Figure 5. Schematic Diagram of Basic RRT Algorithm Extension

The flowchart of the basic RRT algorithm is shown in Figure 6.

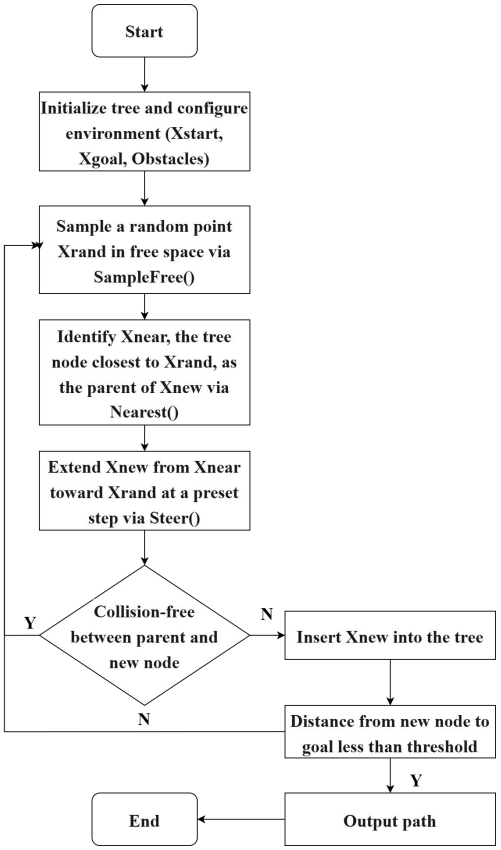


Figure 6. Basic RRT Algorithm Expansion Flowchart

This algorithm rapidly constructs a search tree in three-dimensional space by expanding nodes through random sampling. After initializing the starting point Xstart as the root node, the algorithm enters the main loop: First, generate a random sampling point Xrand within the environment. Next, retrieve the nearest node Xnear to Xrand in the search tree. Finally, generate a new node Xnew along the direction from Xnear to Xrand using a fixed step size Step. After obtaining Xnew, perform collision detection. If any collision is detected, return "Collision," discard this expansion, and proceed to the next sampling round. Otherwise, add Xnew to the tree node list. Iteration terminates upon reaching the target region. SampleFree () is the sampling function, Nearest () locates the nearest node to Xrand in the random tree, and Steer() generates Xnew.

4. Improved RRT Algorithm

4.1 Neighborhood Search

Neighborhood search begins during the sampling phase by first identifying all existing tree nodes Xi within a spherical region centered at the new node Xnew with radius R, forming

the candidate reconnection set PotentialParent. Subsequently, through parent pointer backtracking, all ancestor indices are established and collected into the set Index. Finally, a set difference operation is performed on PotentialParent to remove members common to both PotentialParent and Index. This process retains only non-ancestral nodes within the neighborhood in PotentialParent, ensuring subsequent reconnections occur exclusively on non-ancestral nodes and thereby preventing loop formation.

The neighborhood radius R determines how many existing nodes participate in reconnection during "sub-reconstruction." However, selecting a larger R is not necessarily better. While a larger R increases the number of candidate points and theoretically yields shorter, smoother paths, it also causes computational complexity and collision detection frequency to increase linearly. Furthermore, it may introduce distant, high-cost edges that dilute the effects of local optimization. Experiments indicate that setting R to five times the step size (R=5*Step) yields optimal path planning performance. At this value, the number of nodes within the neighborhood precisely covers the primary branches of the current expansion surface. This approach fully leverages local information while avoiding redundant computations and memory overhead caused by excessive spans. Consequently, the autonomous parking robot demonstrates superior path length, smoothness, and real-time performance in intelligent garage scenarios compared to settings with larger or smaller R values.

As shown in Table 1, the impact of R on the performance of the improved algorithm in this paper was investigated across three metrics—average path length, average time, and success rate—under a simple single-target-point scenario.

Table 1. The Impact of Neighborhood Range R on Algorithm Performance

R	Average length/m	Average time spent/s	Success rate
2**Step*	23.56	13.57	58%
4**Step*	19.72	9.85	100%
5**Step*	13.54	6.71	100%
6**Step*	16.88	9.07	100%
8**Step*	20.35	11.56	97%

After 100 repeated experiments, optimal performance was achieved when the neighborhood radius R was set to 5*Step. Therefore, R=5*Step is recommended as the optimal connection radius for parking robot operation scenarios.

4.2 Target Bias Strategy

The target bias strategy employs a sampling function that incorporates a sub-reconstruction strategy to select sampling points. The sampling function is defined as in Equation (2):

$$X_{\text{randx}} = \begin{cases} X_{\text{goal}} & ,P \le a \\ X_i & ,a < P < b \\ X_{\text{rand}} & ,P \ge b \end{cases} \quad (2)$$

In the formula: Xrand represents the random sampling point; P is a random number between 0 and 1; a and b (a < b) denote the target sampling probability and random sampling probability, respectively. First, generate Xrand on the map; second, select the method for generating Xrandx based on the relationship between P, a, and b. The smaller |a-b| is, the more likely Xgoal or Xrand will be selected as the sampling point. Conversely, the larger |a-b| is, the more likely a point Xi within the neighborhood range R of Xnew will be chosen as a child node for path expansion, i.e., the child reconstruction strategy.

When selecting Xi, first perform cumulative cost evaluation followed by integrated cost calculation. Cumulative cost evaluation is expressed as in Equation (3):

$$c(X_i) > c(X_{\text{new}}) + \left\| X_{\text{new}} - X_i \right\| \quad (3)$$

In the formula: Xi represents a candidate node within the neighborhood, c(X) denotes the cumulative cost from the starting point to node X, and ||X−Y|| is the Euclidean distance between points X and Y. If the cumulative cost is less than the original cost of Xi, it has the opportunity to become a child node of Xnew.

The target bias strategy employs a distance-weighted approach to compute the integrated cost for nodes within neighborhood R. To comprehensively account for both the distance between node Xi and the target point and the distribution characteristics of the tree structure, the distance-weighted method is selected for calculation. The weighting coefficients are dynamically adjusted based on factors such as the target distance. Specifically, for each candidate node Xi within neighborhood R, its integrated cost is calculated as shown in Equation (4):

$$C_i = \begin{cases} 0.8 \cdot D_{\text{new}}(X_i) + 0.2 \cdot D_{\text{goal}}(X_i), 1 \ge w_1(s) > 0.8 \\ w_1(s) \cdot D_{\text{new}}(X_i) + w_2(s) \cdot D_{\text{goal}}(X_i), 0.8 \ge w_1(s) \ge 0.2 \\ 0.2 \cdot D_{\text{new}}(X_i) + 0.8 \cdot D_{\text{goal}}(X_i), 0.2 > w_1(s) > 0 \end{cases} \quad (4)$$

$$D_{\text{new}}(X_i) = \| X_i - X_{\text{new}} \| = \sqrt{(x_i - x_{\text{new}})^2 + (y_i - y_{\text{new}})^2 + (z_i - z_{\text{new}})^2} \quad (5)$$

$$D_{\text{goal}}(X_i) = \| X_i - X_{\text{goal}} \| = \sqrt{(x_i - x_{\text{goal}})^2 + (y_i - y_{\text{goal}})^2 + (z_i - z_{\text{goal}})^2} \quad (6)$$

In the formula: Ci represents the integrated cost of Xi; w1(s) and w2(s) denote the weight coefficients for the distance between Xi and the new node and the target point, respectively; Dnew (Xi) indicates the Euclidean distance between Xi and Xnew; Dgoal (Xi) indicates the Euclidean distance between Xi and the target point. In the early phase, w1(s) is relatively large, and Dnew (Xi) has a greater influence on Ci, allowing for thorough directional exploration around Xnew. In the later phase, w2(s) is relatively large, and Dgoal (Xi) has a greater influence on Ci, enabling rapid convergence toward the target point and enhancing directionality. Nodes with lower integrated costs are more likely to be selected as child nodes. This achieves automatic adjustment of the proportion of paths in both parts, making the integrated cost function more valuable for reference.

The dynamic adjustment mechanism refers to: during path expansion, the weights w1(s) and w2(s) are dynamically adjusted based on changes in the distance between Xnew and Xgoal, thereby achieving a balance between exploration and convergence. During the early search phase, the system favors increasing w1(s) to enhance tree expansion and diversity. In the later search phase, as the tree gradually approaches the target region, the weight of w2(s) is elevated to strengthen goal orientation, accelerating the path's convergence toward the target point. This dynamic strategy effectively balances global exploration and local convergence, significantly improving expansion efficiency.

As shown in Equations (7) and (8), w1(s) and w2(s) are computed based on the distance s to achieve dynamic adjustment:

$$w_1(s) = \begin{cases} \frac{s_{\text{goal}}}{s}, s_{\text{goal}} < s \\ 0.2 \quad , s_{\text{goal}} \geq s \end{cases} \quad (7)$$

$$w_2(s) = 1 - w_1(s) \quad (8)$$

In the formula: sgoal represents the straight-line distance between Xnew and Xgoal; s denotes the Euclidean distance between Xgoal and Xstar.

The target bias strategy employs sampling functions and cumulative, integrated cost to achieve path exploration. Appropriate selection of a and b not only enhances the diversity required during the early stages of path exploration but also ensures rapid convergence toward the target point in later stages. This minimizes the deviation of Xnew and Xgoal from the target point Xi, aligning them as closely as possible along the same straight line and thereby increasing the efficiency of directional guidance.

4.3 Local Rewiring

Local rewiring involves a two-step "cost-collision" dual evaluation. First, compare the integrated costs one by one among all Xi that satisfy the cumulative cost requirements. Second, select the node with the lowest integrated costs among collision-free reconnections. As shown in Figure 7, the collision assessment employs the center-point safety distance method. Based on the external circle model of the automated parking robot described earlier, the robot is treated as a rectangular prism. The MATLAB point-based robot is considered the center point of this prism. The center point must maintain a safe distance from obstacles without collision. Any collision or failure to maintain the safe distance is deemed a collision.

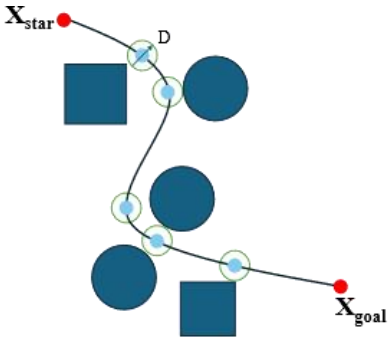


Figure 7. Schematic Diagram of Center Point Safe Distance Method

The specific steps are as follows: When the point robot traverses the path between Xi and Xnew, detect whether there are obstacles within the spherical region centered at the point robot with a diameter equal to the safety distance D. Finally, redirect the parent pointer of the compliant node Xi to Xnew and update the cost c (Xi).

4.4 RRT-CRe Algorithm Flow

The RRT-CRe algorithm flow is shown in Figure 8, with its extension illustrated in Figure 9.

This algorithm employs a sub-reconstruction strategy and target bias strategy to optimize path generation, enhance path planning efficiency, and ensure paths better align with the operational environment of automated parking robots.

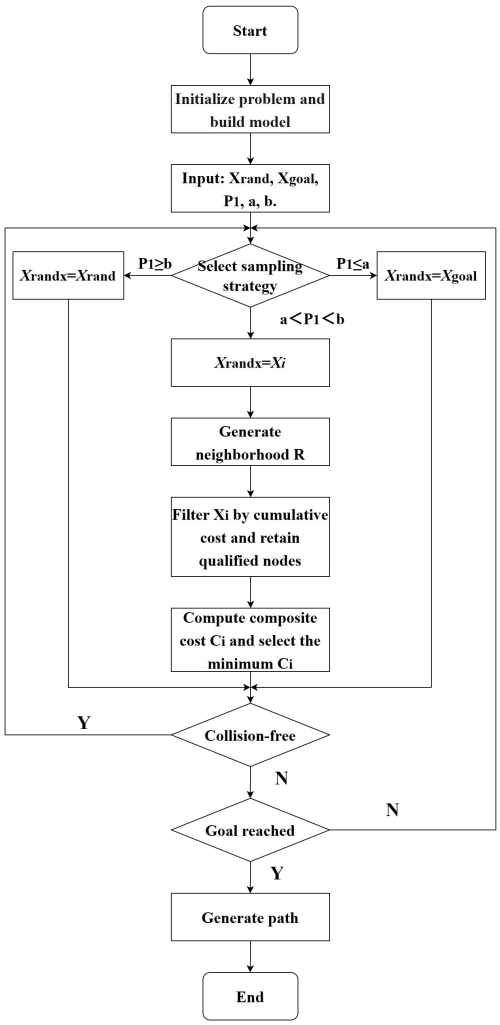


Figure 8. Improved RRT-CRe Algorithm Overall Flowchart

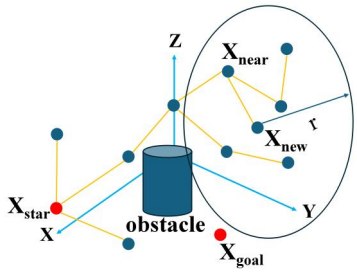


Figure 9. Improved RRT Expansion Diagram

5. Simulation Analysis

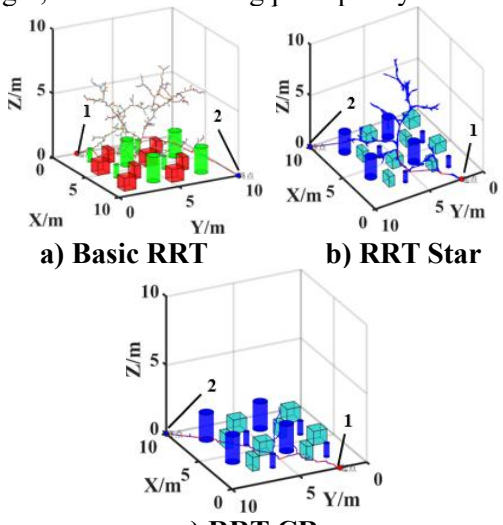
5.1 Environment Configuration

To validate the performance of the RRT-CRe algorithm, a simulation environment was established using MATLAB on a Windows 10 computer. The path planning capabilities of the three algorithms were compared within the previously described environment. The experimental hardware configuration comprised an Intel® Core™ i7-10750H CPU@2.60GHz processor, a 500GB solid-state drive, and 16GB RAM.

5.2 Result Comparison

Simulation comparisons were conducted on the results of the basic RRT algorithm, RRT Star algorithm, and RRT-CRe algorithm under different environments to validate the effectiveness of the RRT-CRe algorithm. The simulation results are shown in Figures 9 to 11. The curved segments in the figures represent redundant paths generated during the algorithm's exploration process, while the straighter segments indicate the final paths determined by the algorithm.

As shown in Figures 10–12, under appropriate parameters, all three algorithms can find a feasible path from the starting point to the target point. Among them, the path generated by the basic RRT algorithm contains a large number of redundant nodes and exhibits overall low quality. The RRT Star algorithm demonstrates a certain improvement in quality compared to the basic RRT algorithm, with reduced redundant nodes and path length in the planned path. Compared to the previous two algorithms, the RRT-CRe algorithm significantly reduces redundant nodes in the path and noticeably shortens the path length, further enhancing path quality.

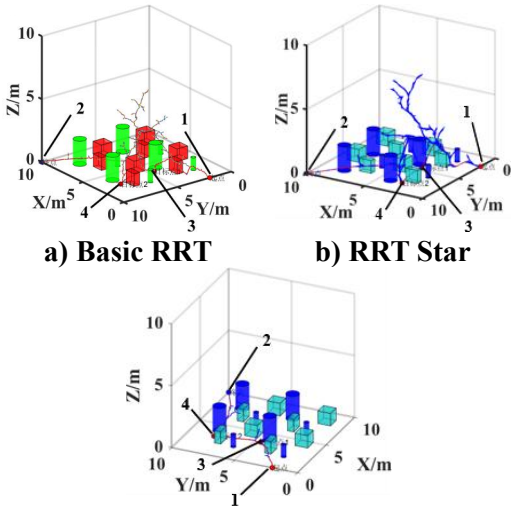


a) Basic RRT **b) RRT Star**

c) RRT-CRe

Figure 10. Experimental Results in a Simple Environment with a Single Target Point

1-Start
2-End

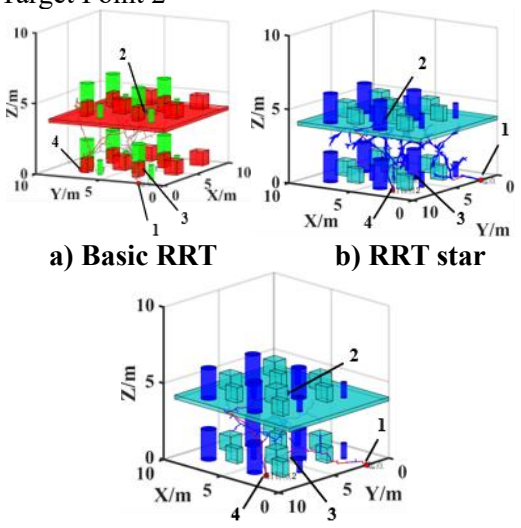


a) Basic RRT b) RRT Star

c) RRT-CRe

Figure 11. Experimental Results in a Simple Environment with Multiple Target Points

1-Starting Point
2-Ending Point
3-Target Point 1
4-Target Point 2



a) Basic RRT b) RRT star

c) RRT-CRe

Figure 12. Experimental Results in Narrow and Complex Environments with Multiple Target Points

1-Starting Point
2-Ending Point
3-Target Point 1
4-Target Point 2

As shown in Tables 2 to 4, the planning results of the three algorithms under the three scenarios mentioned above are compared. Since this study investigates sampling algorithms with randomness, all experiments were conducted under optimal parameter conditions to ensure reliability. Each scenario underwent 1000 repeated experiments, with the average values of each metric serving as the final results.

Table 2. Algorithm Performance Comparison in Simple Environment Single-Target Scenarios

Algorithm	Average Path Length/m	Average time spent/s	Success rate
Basic RRT	19.28	2.71	100%
RRT Star	16.07	0.76	100%
RRT-CRe	15.46	0.79	100%

As shown in Table 2, in the scenario of a single target point in a simple environment, all three algorithms achieved a success rate of 100%. The path lengths generated by the RRT-CRe algorithm and RRT Star were reduced by 19.81% and 16.65%, respectively, compared to the basic RRT. In terms of average time consumption, both the proposed algorithm and RRT Star achieved significant reductions relative to the basic RRT. Although the proposed algorithm's time consumption was comparable to RRT Star in the simple environment with a single target point scenario, it demonstrated substantial improvement over the basic RRT.

As shown in Table 3, in the simple environment with multiple target points, the RRT-CRe algorithm demonstrates significant improvements over both the basic RRT and RRT Star algorithms. In terms of average path length, the RRT-CRe algorithm reduces the average path length by 32.68% and 9.40% compared to the basic RRT and RRT Star algorithms, respectively. In terms of average time, the RRT-CRe algorithm reduced the time by 63.04% and 64.55% compared to the basic RRT and RRT Star algorithms, respectively. In terms of success rate, only the RRT-CRe algorithm achieved a 100% success rate. This demonstrates that the RRT-CRe algorithm exhibits stronger adaptability to multi-objective points than the other two algorithms.

Furthermore, the average time consumption reveals that RRT Star exhibits limited optimization of the global path during the expansion phase due to restricted exploration directions, resulting in a narrow optimization scope and relatively longer execution times. In contrast, RRT-CRe overcomes these associated issues.

As shown in Table 4, the RRT-CRe algorithm also demonstrates superior performance in scenarios involving multiple target points within narrow, complex environments.

In terms of average time consumption, the RRT-CRe algorithm reduces the time required

by 56.78% and 31.99% compared to the basic RRT algorithm and RRT Star algorithm respectively, significantly accelerating path planning speed under multi-target point conditions in narrow, complex environments. In terms of success rate, the RRT-CRe algorithm achieved 100%, significantly outperforming the other two algorithms. This demonstrates that in scenarios resembling intelligent garage environments—characterized by narrow, complex spaces and multiple target points—the RRT-CRe algorithm enables automated parking robots to efficiently handle path planning challenges.

Table 3. Algorithm Performance Comparison in Simple Environment Multi-Target Scenarios

Algorithm	Average Path Length/m	Average time spent/s	Success rate
Basic RRT	26.07	2.57	61.5%
RRT Star	19.37	2.68	90.2%
RRT-CRe	17.55	0.95	100%

Table 4. Performance Comparison of Algorithms in Narrow and Complex Multi-Target Environments

Algorithm	Average Path Length/m	Average time spent/s	Success rate
Basic RRT	31.14	5.46	55.1%
RRT Star	28.85	3.47	65.4%
RRT-CRe	26.64	2.36	100%

6. Conclusion

Through in-depth research on the Rapidly-exploring Random Tree (RRT) algorithm, this study addresses the inherent limitations of the basic RRT algorithm that hinder its direct application to the efficient path planning of automatic parking robots. Specifically, the basic RRT algorithm exhibits excessive randomness, which tends to generate a large number of invalid nodes in narrow corridors and multi-obstacle scenarios, resulting in low sampling utilization. To mitigate these issues, this paper improves the sampling and expansion phases of the basic RRT algorithm, proposes a goal-biased strategy optimized by a sub-reconstruction mechanism, and adapts the collision detection step to the specific operating environment of automatic parking robots.

Simulation experiments conducted in MATLAB demonstrate that the RRT-CRe algorithm achieves a significant reduction in planning time compared to the basic RRT algorithm in simple single-target-point environments. Furthermore, the RRT-CRe algorithm outperforms the RRT* algorithm in terms of planning success rate in narrow, complex multi-target-point environments. These results indicate that the RRT-CRe algorithm possesses superior adaptability to complex scenarios compared to both the basic RRT and RRT* algorithms. It enhances the purposefulness of the sampling process to a certain extent, playing a crucial role in improving the path planning success rate and reducing planning time for automatic parking robots in intelligent parking garages. Thus, this study provides a more lightweight, efficient, and reliable solution for path planning technology.

References

[1] Wang L, Zhu X, Li Z, et al. Ultrasonic Obstacle Avoidance and Full-Speed-Range Hybrid Control for Intelligent Garages. Sensors, 2024, 24(17):5694-5696.

[2] Jiang N, Han Y. Patent technology development of intelligent stereo garage. China Science and Technology Information, 2024, (14): 31–33.

[3] Neeru M, Mamta M, Jude D H, et al. Deep learning and saliency-based parking IoT classification under different weather conditions. Intelligent Decision Technologies, 2024, 18(2):1411-1424.

[4] Venkata S, Sai M, Yaswanth T, et al. Smart garage utilizing Internet of Things (IoT). Journal of Sensors, 2022, 1156(22): 255-265.

[5] Ren Z, Cai A, Xu F. Automated guided vehicle (AGV) path optimization method based on improved rapidly-exploring random trees. Computer Science, 2025, 11: 15-24.

[6] Zhao X, Wang K, Zhang P, et al. An improved RRT* path planning algorithm combining Gaussian distributed sampling and depth strategy for robotic arm of fruit-picking robot. Computers and Electronics in Agriculture, 2025, 207: 111244.

[7] Venu S, Gurusamy M. A comprehensive review of path planning algorithms for autonomous navigation. Results in Engineering, 2025, 28: 107750.

[8] Jeong I, Lee S, Kim J. Quick-RRT*: Triangular Inequality-based Implementation

of RRT* with Improved Initial Solution and Convergence Rate. Expert Systems with Applications, 2019, 123:82-90.

[9] Bin L, Yi H, Fang W, et al. Stack-RRT*: A Random Tree Expansion Algorithm for Smooth Path Planning. International Journal of Control, Automation and Systems, 2023, 21(3): 993-1004.

[10] Wang H, Zhou X, Li J, et al. Improved RRT* algorithm for disinfecting robot path planning. Sensors, 2024, 24(5): 1417.

[11] Jun D, Yin Z, Xia H, et al. An improved RRT* algorithm for robot path planning based on path expansion heuristic sampling. Journal of Computational Science, 2023, 67: 102085.

[12] Wu Z, Dai J, Jiang B, et al. Robot path planning based on artificial potential field with deterministic annealing. ISA Transactions, 2023, 138:74–87.

[13] Yang Z, Hu J, Zhao H. AMP-RRT*: an adaptive multi-layer path planning algorithm for robots in complex environments. Engineering Research Express, 2025, 7(3): 12–19.

[14] Kiani F, Seyyedabbasi A, Aliyev R, et al. Adapted-RRT: novel hybrid method to solve three-dimensional path planning problem using sampling and metaheuristic-based algorithms. Neural Computing and Applications, 2021, 33(24): 17129–17159.

