Trajectory Planning Research Based on Rapidly-exploring Random Trees (RRT) Algorithm and Modified Forms

Yuxuan Lan *

School of Mechanical Engineering, Xi'an Jiaotong University, Xi'an, China

* Corresponding Author Email: YuxuanLan@stu.xjtu.edu.cn

Abstract. With the rapid advancement of computer technology, robots have become indispensable in various fields. Trajectory planning plays a crucial role in optimizing robot performance by enhancing work efficiency and reducing operational costs. While the widely used Rapidly-exploring Random Trees (RRT) algorithm is effective in path planning, it often generates non-optimal paths with poor smoothness. By incorporating elements from RRT-Connect and reverse optimization, the algorithm can efficiently navigate through intricate and dynamic surroundings. Furthermore, the utilization of cubic polynomial interpolation in optimizing path curves adds a layer of sophistication to the trajectory planning process. This interpolation technique allows for the creation of smoother and more natural path trajectories, reducing jerky movements and enhancing overall motion control precision. The paths produced by the enhanced RRT algorithm demonstrate increased continuity and visual attractiveness, crucial for tasks necessitating accurate and elegant robotic motions. The improved algorithm showcases reduced path length and time consumption, as well as enhanced path smoothness. Particularly, relative to the traditional RRT approach, the time consumption has dropped by 69.15% and the median duration of the paths has fallen by 21.11%. These improvements signify a substantial advancement in trajectory planning for robots, offering more efficient and smoother paths for robotic operations across various applications.

Keywords: Trajectory planning; RRT algorithm; reverse optimization; polynomial interpolation.

1. Introduction

Robot trajectory planning uses a planning algorithm to create a path from the beginning point to the destination area that is free of collisions and satisfies the robot's limitations in an obstacle-filled working environment. The quality of trajectory planning algorithms directly affects whether a robot can complete its tasks. Excellent path planning algorithms can reduce the robot's running time and improve its search efficiency. RRT (Rapidly-exploring Random Tree) was proposed by Professor Lavalle in 1998, and in recent decades, a probabilistic sampling-based incremental search algorithm has been extensively developed and utilized [1, 2].

As a sampling-based path planning algorithm, RRT does not require preprocessing of the map, enabling fast searching. However, the RRT algorithm also has its drawbacks, such as relatively blind search, complex computations in complex environments, long running times, and susceptibility to getting stuck in dead ends [3, 4]. This article will conduct research based on RRT and its existing variants, setting specific trajectory planning maps. Using Matlab for simulation experiments to confirm the efficacy of the algorithm, comparing and summarizing the paths generated by different algorithms, and summarizing the regularities. To overcome the limitations of current RRT algorithms, the author integrates the bidirectional Rapidly-exploring Random Tree Connect (RRT-Connect) algorithm, reverse optimization approach, and cubic polynomial interpolation technique to introduce a novel enhanced algorithm aimed at improving the efficiency of path planning.

The second part of the article summarizes and compares existing RRT algorithms and their improved variants, highlighting their respective advantages and disadvantages. Based on this analysis, a new path planning algorithm is proposed, along with an explanation of its implementation logic. Path planning simulations are then conducted, based on MATLAB, in specific application scenarios to validate the effectiveness of various RRT and its variants algorithms. A comparative analysis of the planned results is performed, including the application of cubic polynomial interpolation to smooth the paths generated by the improved algorithm. The results indicate that while all algorithms

successfully plan paths, the generated paths differ in their characteristics. Moving on to the fourth section, further analysis is conducted based on multiple simulations to verify the effectiveness and stability of the algorithms. The simulation results indicate that the optimized RRT algorithm introduced in this research surpasses other algorithms in terms of planning time, path length, and algorithm stability.

2. Methodology

2.1. Introduction to Application Scenarios

As shown in the Figure 1, the factory has a planar layout, with an overall rectangular shape of 80m x 60m, consisting mainly of two parts: a thermostatic chamber and a processing workshop. The main task of the mobile robots is to transport goods from specified starting points to target points. During the transportation process, paths are planned based on missions. The numerical labels shown in Figure 1 represent possible endpoints or starting points for the robots.

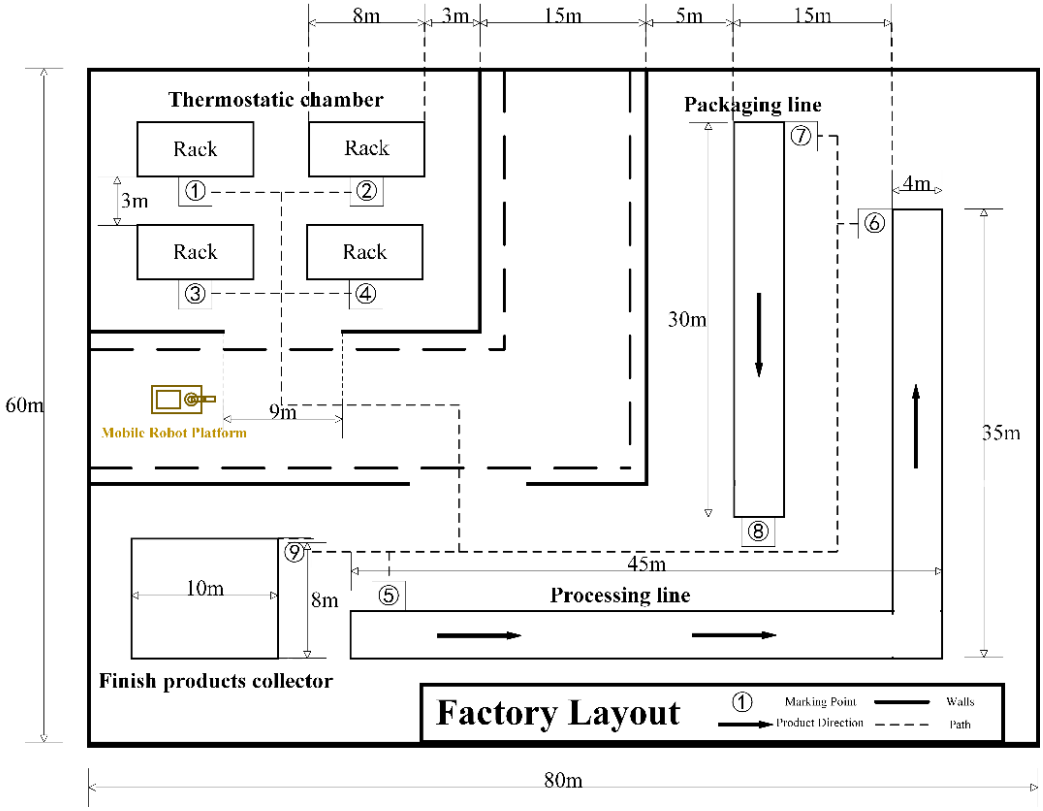


Fig 1. Trajectory planning Factory layout (Photo credited: Original)

2.2. Principle of RRT Algorithm and Modifications

This paper concentrates on investigating path planning utilizing the RRT algorithm and its derivatives. The RRT algorithm is a trajectory planning technique well-suited for intricate and high-dimensional environments. It explores viable paths by employing random sampling and constructing a tree structure. The fundamental concept of the RRT algorithm involves commencing from the starting point, creating a sequence of random points via random sampling, and subsequently linking these points to the current tree structure, forming a tree-shaped path exploration area. During the connection process, the presence of obstacles is taken into accounts, and only paths that do not intersect with obstacles are added to the tree. By continuously generating and connecting random points, the algorithm can quickly traverse the entire search space and identify a viable path. Figure 2 shows the pseudocode of the algorithm:

RRT Algorithm

Input: M,x_{init}, x_{goal}
Result: A path Γ from x_{init} to x_{goal} τ.
$\Gamma.init()$;

for $i = 1$ to n do
  $x_{rand} \leftarrow Sample(\mathcal{M})$:
  $x_{near} \leftarrow Near(x_{rand}, \mathcal{T})$;
  $x_{new} \leftarrow Steer(x_{rand}, x_{near}, StepSize)$;
  $E_i \leftarrow Edge(x_{new}, x_{near})$;
  if $CollisionFree(\mathcal{M}, E_i)$ then
    $\mathcal{T}_{.addNode}(x_{new})$;
    $\mathcal{T}.addEdge(E_i)$;
  if $x_{new} = x_{goal}$ then
    Success();

Fig 2. RRT algorithm pseudo code [2]

M represents the map environment, x_{init} is the initial position, and x_{goal} is the target position. The process of searching the path space starts from the starting point. It randomly scatters a point x_{rand}, then finds the node x_{near} closest to x_{rand}, and moves forward in the direction from x_{near} to x_{rand} by step size to obtain x_{new}. The CollisionFree (M, E_i) method checks if the Edge (x_{new}, x_{near}) intersects with obstacles. If there is no collision, one spatial search expansion is successfully completed. This procedure is iterated until the target position is attained.

In addition to the basic RRT algorithm, this article also utilizes several improved variants of the algorithm for path planning on the same map settings, including RRT* and RRT-Connect. RRT* algorithm is an improvement and extension of the RRT algorithm, aiming to enhance path quality and search efficiency. The RRT* algorithm introduces optimization steps to find better paths and explore the search space more uniformly. However, in the RRT* algorithm, when connecting new sampled points to the tree structure, it considers the cost of the trajectory (such as path length) and selects the path with the minimum cost for connection. This gradual optimization of paths leads to finding paths closer to the best possible solution.

RRT-Connect algorithm aims to find a feasible path between two given states [5, 6]. The core concept of the RRT-Connect algorithm involves the concurrent construction of two trees: one originating from the initial state and the other originating from the goal state. The two trees grow gradually by random sampling and connecting nodes until the nodes of the two trees meet. During the growth process, The RRT-Connect algorithm considers the existence of obstacles, and only incorporates paths that do not intersect with obstacles into the trees. When the nodes of the two trees meet, the RRT-Connect algorithm checks if there exists a path that can connect the initial state and the goal state. If such a path exists, a feasible path is obtained. Table 1 provides a comparison of the Pros. and Cons. of the three algorithms mentioned above [7, 8].

Drawing from the outlined algorithms, this paper introduces an enhanced version of the RRT algorithm, merging the RRT-Connect algorithm with a reverse optimization strategy to attain superior planned paths and reduced computational time, denoted as RRT-Connect-Reverse (abbreviated as RRT-CR). The fundamental concept of the RRT-CR algorithm is to swiftly create a viable path linking the start and end points through the utilization of RRT-Connect.Then, through reverse optimization, the path is optimized to obtain the shortest path. Reverse optimization leverages the concept of the shortest line segment between two points to shorten the path length [9]. In Figure 3, point O represents the starting point, point F represents the end point. The nodes generated by the original RRT algorithm expansion are labeled as A, B, C, D, E. Obstacles are represented by black rectangles. The path obtained after modification is represented by the solid line between O and F, whereas the dotted line indicates the preliminary path produced through the original RRT-CR algorithm.

Table 1. Comparison of RRT and its improved algorithms

Algorithm	Pros.	Cons.
RRT	Few parameters, simple structure, strong search capability, easy integration with other algorithms	Low node utilization in the later stages, high computational complexity, and unstable paths
RRT*	Asymptotic optimality, can reach the best possible solution	Node computations are intensive, leading to high memory consumption
RRT-Connect	Greedy algorithm, bidirectional connection, fewer samples, faster convergence speed	Unable to obtain the optimal solution for the planned path

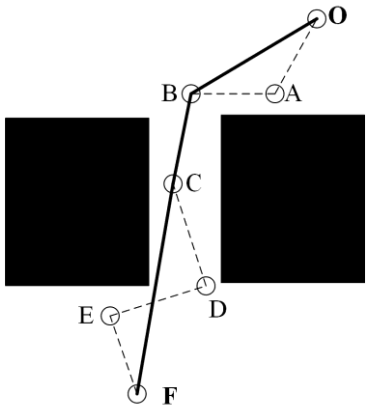


Fig 3. Reverse optimization of path planning (Photo credited: Original)

The detailed optimization process is as follows: Initially, directly connect the starting point O to the end point F. Then, check if the line between O and F is blocked by obstacles. If it is not blocked by obstacles, then the optimal path for this search is the line between points O and F. If it is blocked by obstacles, then following the node order, find the parent node E of F. Check if the line between O and E is blocked by obstacles. If the line between O and E is not clocked by obstacles, then directly consider E as a child node of O, and connect O-E-F successively, forming this path as the optimal path for the search. If the line between O and E is blocked by obstacles, continue to search for the parent node of E, which is D in the figure, and check if the line between the starting point O and D is blocked by obstacles. Following this method continuously, we optimize the path and obtain B as a child node of O. Then, take B as the starting point for the next algorithm modification step, repeating the above steps. Once the complete path is optimized, the optimal path is achieved, and the algorithm terminates.

The RRT-CR algorithm combines the fast speed of RRT-Connect with the ability to obtain the shortest path through reverse optimization, resulting in planned paths that are both fastest and shortest. The algorithm outperforms RRT* and RRT-Connect.

3. RRT Algorithm and Variants Programing

3.1. Algorithm Program Implementation

Utilize MATLAB to implement the RRT, RRT*, RRT-Connect, and RRT-CR algorithms, enabling a robot to navigate from the designated start to end points within the provided scenario. First, abstract the layout of the factory as shown in Figure 4, where black rectangles represent obstacles such as walls, equipment, or debris. Based on the factory layout schematic, set two possible sets of start and end points: ① to ⑤ and ① to ⑦.

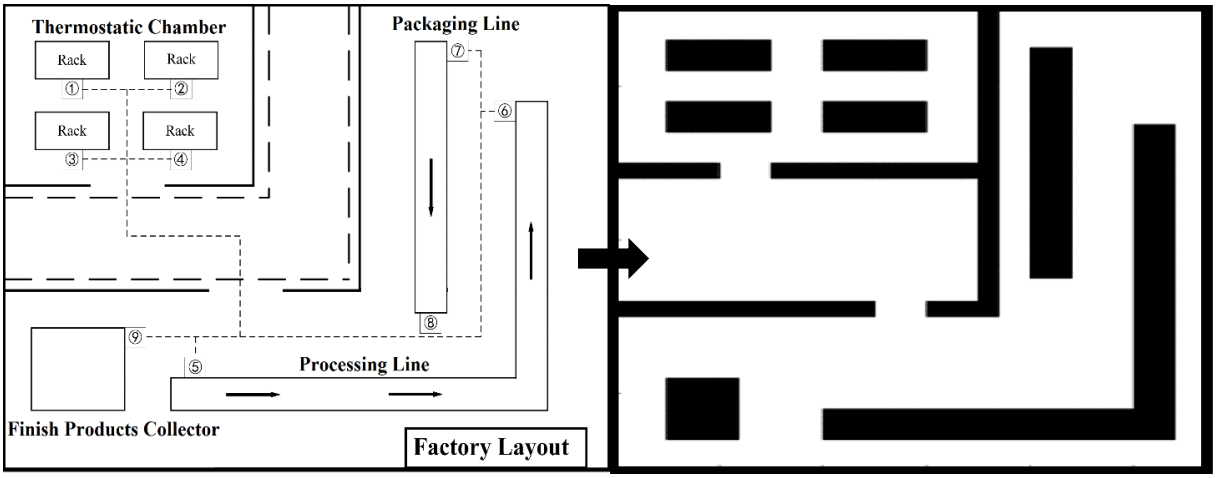
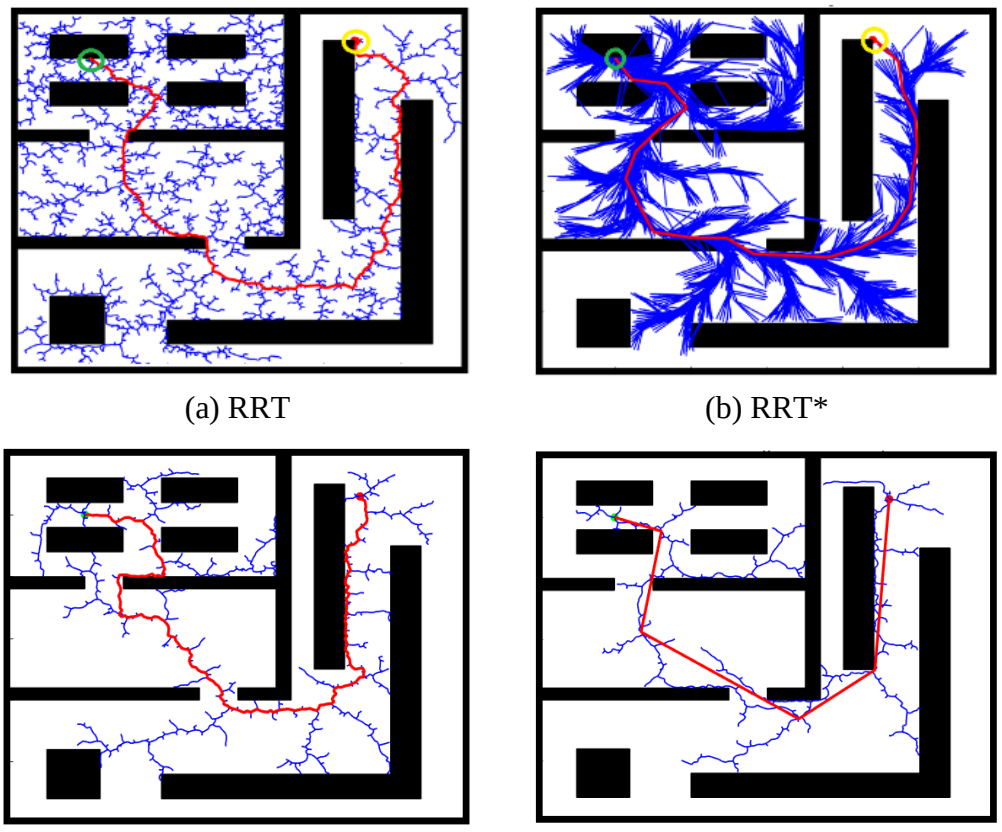


Fig 4. Two or more references (Photo credited: Original)

Comparing RRT-CR algorithm with existing algorithms, implement simulation experiments on the same obstacle layout in a two-dimensional layout to generate paths for two sets of start and end points: ① to ⑤ and ① to ⑦. Set the random step size to 0.5, search radius to 1, and maximum iteration count to 10000. The following are path diagrams obtained from simulation experiments of several algorithms from start point ① to goal point ⑦, shown in Figure 5. It can be observed that all four algorithms can generate feasible paths between the start and goal points. However, RRT and RRT-Connect generate very rough paths with many sharp corners. RRT* can generate relatively shorter paths but is still not optimal, and the planning time for RRT* algorithm is the longest. Conversely, the RRT-CR algorithm is capable of producing the most concise path connecting the start and end points.



(a) RRT (b) RRT*

(c) RRT-Connect (d) RRT-CR

Fig 5. RRT, RRT*, RRT-Connect, RRT-CR algorithm path planning (Photo credited: Original)

3.2. Trajectory Smoothing

The shortest path between the start and target sites can be generated by the RRT-CR algorithm, although it is frequently not a smooth one, according to Figure 5. Robot needs to decelerate at nodes and then accelerate again, which increases the robot's energy consumption and working time. To make the path smoother and allow the robot to move smoothly, the RRT-CR algorithm is modified by using cubic polynomial interpolation. Cubic polynomial interpolation is a commonly used interpolation smoothing algorithm that generates a continuous and smooth curve through a series of control points. Cubic polynomial interpolation introduces the requirement of continuous first and second derivatives of the curve. The entire interpolation interval is divided into several small segments, with each segment using a cubic polynomial for interpolation. The cubic polynomial for each small segment is determined by four control points, where two control points are adjacent data points, and the other two control points are obtained by solving a tridiagonal linear equation system [10, 11]. The algorithm for generating the planned path is named RRT-CR-Smooth (abbreviate to RRT-CRS). The path planned by RRT-CRS is depicted in Figure 6, exhibiting a notably smoother trajectory in contrast to the path generated by the RRT-CR algorithm.

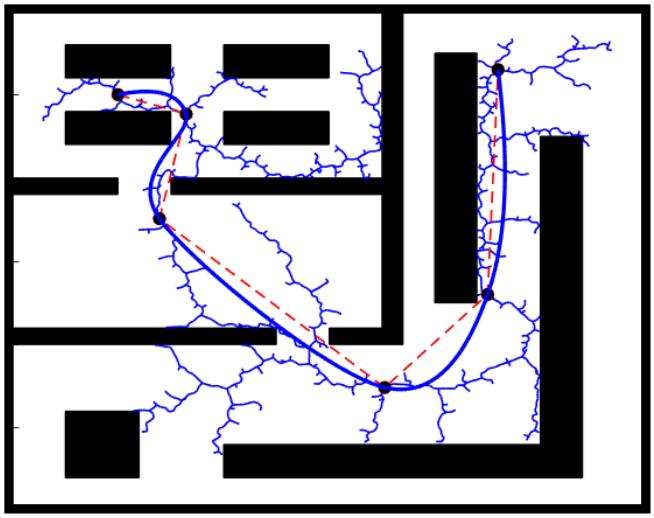
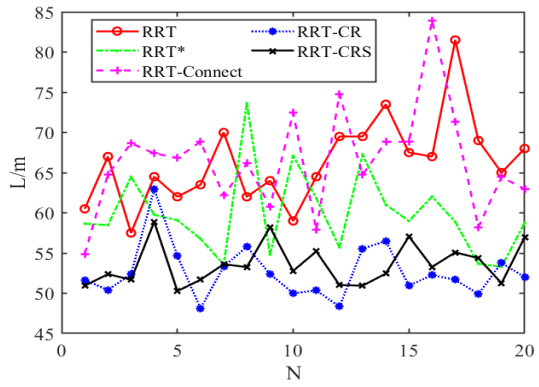


Fig 6. RRT-CRS algorithm path planning result (Photo credited: Original)

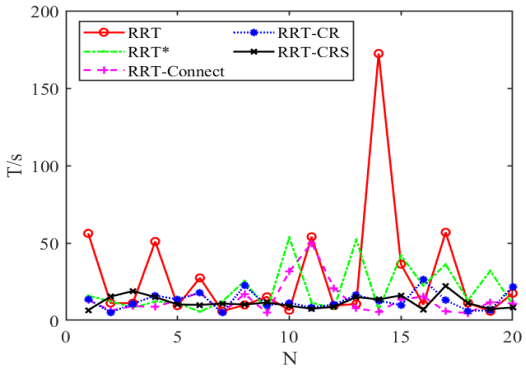
4. Simulation Experiment Result Analysis

4.1. Results of Trajectory Planning Algorithms

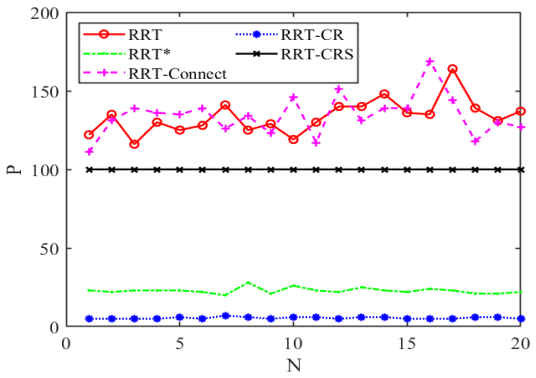
To further demonstrate the improvement effect of the algorithm, 20 comparative experiments were conducted. The data for each experiment, including the path planning time T/s, path length L/m, number of sample points S, and number of path points P, are shown in Figure 7 and Figure 8. The path lengths generated by RRT and RRT-Connect are significantly longer than those generated by RRT-CR and RRT-CRS. Furthermore, the stability of the RRT-CR and RRT-CRS path planning algorithms surpasses in terms of path length, algorithm execution time, and the quantity of sampled points. In 20 repeated experiments, the simulation outcomes of the optimized algorithm tend to stabilize, whereas the results of the RRT and RRT-Connect algorithms show significant fluctuations. Furthermore, due to the integration of bidirectional random trees and reverse optimization principles, the number of sampled points and path points required by the path planning algorithm are greatly reduced. The RRT-CR and RRT-CRS algorithms require significantly fewer sampled points and path points for trajectory planning compared to other RRT algorithms.



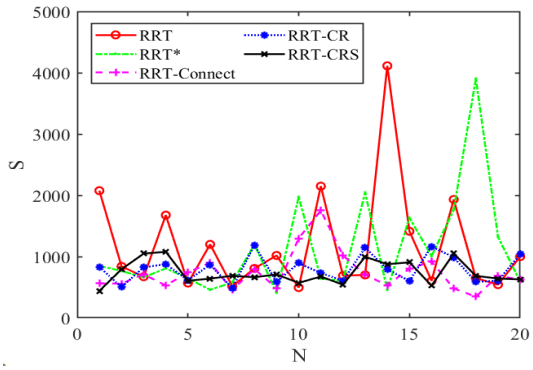
(a) Path Length/m



(b) Time-Consuming/s

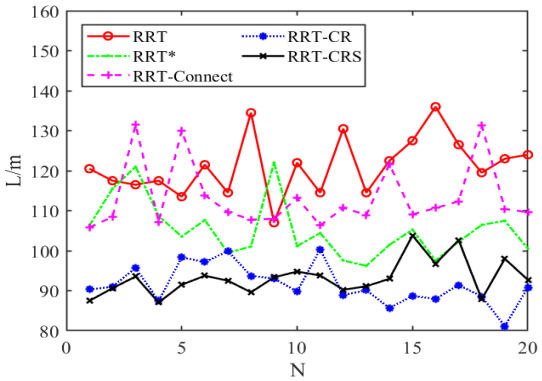


(c) Path Points Number

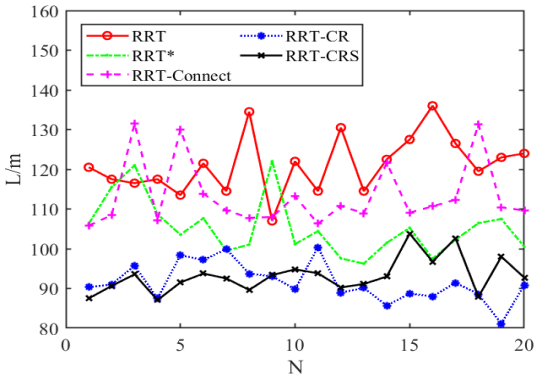


(d) Sample Points Number

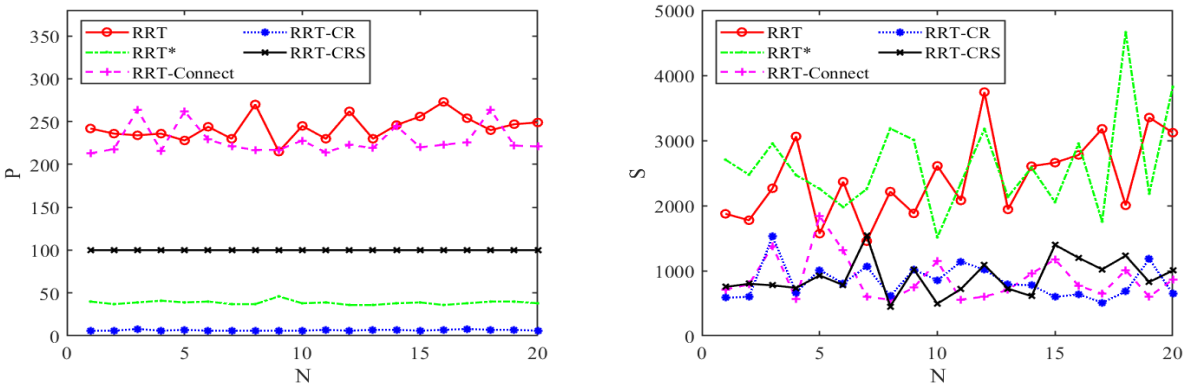
Fig 7. Simulation experiment data of different algorithms (Mission 1: ①>>⑤) (Photo credited: Original)



(a) Path Length/m



(b) Time-Consuming/s



(c) Path Points Number (d) Sample Points Number

① **Fig 8.** Simulation experiment data of different algorithms (Mission 2: ①>>⑦) (Photo credited: Original)

For more convinced comparison of the advantages of the modified algorithm, the experimental data for each group were processed. The data information acquired is displayed in Table 2, where the subscript σ represents the standard deviation, serving as an indicator of the stability of the paths generated by different algorithms.

Table 2. Simulation experiment data of different algorithms

Algorithm	Average T/s	T_σ	Average L/m	L_σ	Average P	P_σ	Average S	S_σ
Start Point ① >> End Point ⑤								
RRT	29.42	5.25	66.25	37.34	133.5	10.50	1182	853.2
RRT*	19.82	14.89	59.88	5.039	22.85	1.796	1115	814.0
RRT-Connect	13.64	10.55	66.44	6.343	134.2	12.65	744.7	317.5
RRT-CR	12.82	5.756	52.64	3.267	5.500	0.591	796.4	217.8
RRT-CRS	11.75	4.140	53.56	2.515	100.0	0.000	737.7	185.7
Start Point ① >> End Point ⑦								
RRT	69.07	31.57	121.1	7.113	243.3	14.22	2426.6	616.5
RRT*	94.47	38.41	105.2	6.990	38.70	2.215	2623.2	711.9
RRT-Connect	14.44	8.116	113.3	8.120	228.1	16.24	875.7	334.1
RRT-CR	12.96	6.986	91.50	4.760	6.550	0.668	835.5	255.3
RRT-CRS	15.02	8.207	93.20	4.333	100.0	0.000	904.5	277.4

4.2. Summary of Path Planning Algorithms Simulation Experiment

The comparison reveals that the RRT-CRS algorithm has made significant advancements in terms of time consumption, average path length, number of sample points, and number of path points. Compared to the RRT, RRT*, and RRT-Connect algorithms, time-consuming has reduced by 69.15%, 62.14%, and 10.13% respectively. The average path length generated by the RRT-CRS algorithm measures 53.57m and 93.2m, showcasing a reduction of 21.11%, 11.50%, and 18.57% in comparison to the paths planned by the remaining three algorithms. In the RRT-CRS algorithm, there are significantly fewer sample points., thereby reducing the generation of useless sample points and making the algorithm converge faster. The number of final path points planned is also significantly reduced. Through comparative simulations, it is evident that the RRT-CRS algorithm has higher search efficiency, shorter planned path length, can ensure real-time performance of the algorithm, and significantly modified the quality of the trajectory planning.

5. Conclusions

The paper introduces a novel path planning algorithm that integrates RRT-Connect and reverse optimization, built upon research carried out on the RRT trajectory planning algorithm. Trajectory planning algorithm simulations are conducted in designated application scenarios, demonstrating that the proposed RRT-CRS algorithm adeptly resolves challenges like prolonged algorithm processing time and an elevated number of sampled points. The paths produced by the algorithm adequately fulfill the motion criteria of robots. To summarize, the research outcomes can be encapsulated as follows:

(1) The article proposes a new RRT trajectory planning algorithm, named RRT-CRS, by combining RRT-Connect with the idea of reverse optimization.

(2) By employing cubic polynomial interpolation optimization in the RRT-CR, the smoothness of the planned path generated by the RRT-CR algorithm is improved.

(3) Based on MATLAB, conducted simulation experiments. In the set map layout, the RRT-CRS algorithm outperformed the other three algorithms. The path length, time-consuming, number of nodes, and sample points were all significantly modified.

The planning of robot trajectories is a crucial element in the realm of autonomous robot navigation. The path planning algorithm introduced in this paper can lay a robust groundwork for forthcoming robot control systems.

Referencess

[1] Zhu, Hu, Li, et al. A Review of Autonomous Driving Vehicle Path Planning Algorithms. Agricultural Equipment and Vehicle Engineering, 2023, 61(11): 18-22.

[2] Chen, Jiang, Zheng. A Review of Rapidly-exploring Random Tree Algorithm for Robot Path Planning. Computer Engineering and Applications, 2019, 55(16): 10-17.

[3] Huang. Research on Mobile Robot Path Planning Based on Improved Sampling Algorithm. Guangxi: Guangxi University, 2021.

[4] Mohamed, Milan. Sampling-based robot motion planning: A Review. IEEE Access, 2014, 2(1):56-77.

[5] Kang, Lim, Choi, et al. Improved RRT-connect algorithm based on triangular inequality for robot path planning. Sensors, 2021, 21(2), 333.

[6] Zafar, Mohanta. Methodology for path planning and optimization of mobile robots: A Review. International Conference on Robotics and Smart Manufacturing (Rosma2018), 2018, 133, 141-152.

[7] Liu. Research on Autonomous Mobile Robot Path Planning Algorithms. Hunan: Changsha University of Science and Technology, 2018.

[8] Shao, Zhi, Hu. The Application of Improved RRT Algorithm in Path Planning. Journal of Nanyang Normal University, 2024, 23(1): 58-63.

[9] Han. Research on Path Planning Based on Improved RRT Algorithm. Mechanical Engineering and Automation, 2023(6): 31-33.

[10] Sidobre, Daniel, Desornneaux, et al. Smooth Cubic Polynomial Trajectories for Human-Robot Interactions. Journal of Intelligent & Robotic Systems: Theory & Application, 2019, 95(3/4): 851-869.

[11] Shi, Xu, Sheng. Obstacle Avoidance Planning with RRT Based on Seventh-degree Polynomial Interpolation. Light Industry Machinery, 2022, 40(6): 1-6.