IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received October 8, 2021, accepted October 23, 2021, date of publication October 27, 2021, date of current version November 2, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3123622

Improved RRT-Connect Based Path Planning Algorithm for Mobile Robots

JIAGUI CHEN[1], YUN ZHAO[1], AND XING XU[2]
[1]School of Information and Electronic Engineering, Zhejiang University of Science and Technology, Hangzhou 310023, China
[2]School of Mechanical and Energy Engineering, Zhejiang University of Science and Technology, Hangzhou 310023, China

Corresponding author: Xing Xu (xuxing@zust.edu.cn)

This work was supported in part by the National Key Research and Development Program of China under Grant 2019YFE0126100, in part by the National Natural Science Foundation of China under Grant 61605173 and Grant 61403346, and in part by the Natural Science Foundation of Zhejiang Province under Grant LY16C130003, and in part by the Science and Technology Project of Zhejiang Province under Grant 2019C54005.

ABSTRACT Path planning plays a key role in the application of mobile robots and it is an important way to achieve intelligent mobile robots. Traditional path planning algorithms need to model environmental obstacles in a deterministic space, which is complex and easily trapped in local minimal. The sampling-based path planning algorithm performs collision detection on the environment and it is able to quickly obtain a feasible path. In order to solve the problem of inefficient search of the sampling-based Rapidly Expanding Random Tree (RRT-Connect) path planning algorithm, an improved RRT-Connect mobile robot path planning algorithm (IRRT-Connect) is proposed in this paper. In order to continue to speed up the search of the algorithm, a simple and efficient third node is generated in the configuration space, allowing the algorithm to be greedily extended with a quadruple tree in the proposed algorithm. Further, the method of adding guidance is proposed to make the algorithm have the characteristics of biasing towards the target point when expanding, which improves the exploration efficiency of the algorithm. In order to verify the effectiveness of the proposed algorithm, this paper compares the execution performance of the four algorithms in six environments of different complexity. The results of the simulation experiments show that the proposed improved algorithm outperforms the RRT, RRT-Connect and RRT* algorithms in terms of the number of algorithm iterations, planning time and final path length in different environments. In addition, the improved algorithm was ported to the ROS mobile robot for experiments with real-world scenarios.

INDEX TERMS Mobile robots, path planning, RRT-connect, target bias, dichotomous method.

I. INTRODUCTION

In recent decades, with the continuous improvement of hardware equipment, mobile robots have been widely used in various fields such as industry, medical care, agriculture, and services [1]–[3]. Path planning is a key technology and a major challenge for mobile robots [4]. Path planning for robots involves searching for an optimal or sub-optimal path from a given start point to a goal point according to certain evaluation criteria (e.g., planning time, path length, etc.). Planning tasks can be divided into global path planning and local path planning depending on whether the environmental information is known or not.

Various algorithms have been developed to solve the path planning problem, such as Dijkstra algorithm [5], which is based on the idea of greed. There are also the heuristic A* algorithm [6], the artificial potential field algorithm [7] and the bionic ant colony algorithm [8] and so on. Some research has refined these basic algorithms to apply them to tasks that suit themselves. Zhang *et al*. [9] improved the heuristic function of the A* algorithm and proposed a new path smoothing strategy in order to generate a safer path further away from obstacles. In complex environments, the authors convert the distance and safety costs of the algorithm into time costs. Gao *et al*. [10] augmented the heuristic ant colony optimization algorithm with four strategies to achieve fast path planning for mobile robots in complex environments. Nazarahari *et al*. [11] combined the path length, smoothness and safety in the path planning problem into a multi-objective path planning problem and proposed an improved artificial potential field algorithm for path planning of multiple mobile robots in continuous

The associate editor coordinating the review of this manuscript and approving it for publication was Heng Wang.

environments. Orozco-Rosas *et al*. [12], [13] investigated a membrane-evolutionary artificial potential field method for solving the path planning problem of mobile robots, which combines membrane computation with an artificial potential field method to find suitable parameters to generate feasible safe paths. There are also studies that speed up the convergence of path planning by introducing the idea of reinforcement learning [14]–[16].

The traditional path planning algorithms mentioned above have an exponential increase in computational complexity in complex environments and the algorithms tend to fall into local minimal. In order to improve the efficiency and practicality of the algorithms, it is necessary to reduce their completeness requirements. Random sampling-based algorithms are commonly used planning algorithms that have probabilistic completeness and resolution completeness respectively, thus reducing the computational complexity of the algorithm.

Among the sampling-based path planning algorithms, one of the most widely used algorithms is the Rapidly exploring Random Trees (RRT) algorithm [17]. The algorithm performs uniform sampling in the configuration space and it is probabilistically complete. As the RRT algorithm samples randomly in space, the paths generated by this algorithm are often not optimal or sub-optimal. A number of studies have improved the RRT algorithm based on its ideas. Zhang *et al*. [18] introduced the regression mechanism into the RRT algorithm in order to address the situation that the RRT algorithm is prone to falling into local minimal in complex environments, and adopted an adaptive node expansion strategy to optimize the direction of new node expansion, avoiding the blind search for robotic arm path planning. Wei and Ren [19] adopted the strategy of directional expansion of new nodes, which is performed for a directional node, which makes the improved RRT algorithm much more efficient and combines curvature constraints for path smoothing.

Some studies have also combined the RRT algorithm with other algorithms to improve the expansion strategy of randomized trees. Xu and Park [20] combined the artificial potential field algorithm with the RRT algorithm to reduce the oscillation phenomenon in the artificial potential field path planning method. Kiani *et al*. [21] proposed an adaptive RRT algorithm that combines three metaheuristics, for Grey Wolf Optimization, Incremental Grey Wolf Optimization, and Expanded Grey Wolf Optimization, respectively. The proposed algorithm eliminates the drawbacks of both the sampling algorithm and the metaheuristic algorithm. In order to enhance the search capability of the RRT algorithm in unknown environments, some studies have incorporated the idea of Q-Learning reinforcement learning for improvement [22], [23].

In order to improve the search speed of spanning trees, Jr and Lavalle [24] proposed the RRT-Connect algorithm for bidirectional exploration, which generates two random trees in the configuration space for expansion from the start to the end and the end to the start, respectively, to further speed up the search efficiency. Although the RRT-Connect algorithm speeds up the expansion of spanning trees, the resulting paths are still not optimal or sub-optimal, and some studies have improved the search efficiency of the RRT-Connect algorithm. Kang *et al*. [25] proposed a triangular inequality based RRT-Connect algorithm using the principle of triangular inequality and experimented in eight environments, reducing the path length by 16% compared to the original RRT-Connect algorithm. In order to obtain better paths, some studies have changed the parent node selection by using a cost function to select the node with the smallest cost in the extended node neighborhood as the parent, so that the algorithm achieves asymptotic optimality [26]–[28]. However, these algorithms iterate slowly at the expense of planning time.

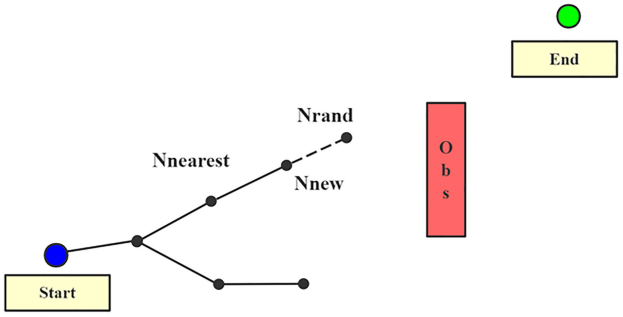

FIGURE 1. RRT algorithm spanning tree expansion process.

Based on the above discussion, in order to speed up the search efficiency of the sampling-based path planning algorithm, this paper proposes an improved RRT-Connect based path planning algorithm, referred to as IRRT-Connect in the following. In general, the contributions to this paper are as follows:

1) In order to speed up the exploration of spanning trees, this paper generates a simple and efficient third node in the configuration space based on the idea of dichotomous points, allowing the algorithm to be extended with four trees.
2) In order to solve the blind search feature of the traditional RRT-Connect algorithm, by increasing the guiding force, the spanning tree is biased towards the direction of the target point every time when expanding new nodes, which speeds up the search efficiency of the algorithm.
3) The performance of multiple path planning algorithms executed in several environments of varying complexity is compared and the proposed algorithms are ported to a real mobile robot to verify the feasibility of the algorithms.

II. RELATED WOEK

This section introduces the relevant background of the thesis, the problem definition, the relevant algorithm flow, etc.

A. RAPIDLY EXPLORING RANDOM TREES (RRT) ALGORITHM

The Rapidly exploring Random Trees path planning algorithm was first proposed by Professor Lavalle. The algorithm is a sampling-based path planning algorithm, which has been widely studied and applied since its introduction due to its simplicity and fast search capability, and its adeptness at solving path planning problems in multi-dimensional spaces and complex environments. RRT constructs a path search tree by incrementally generating a random series of sample points in Configuration Space [29] with the starting point as the root node. Fig. 1 shows the path planning process of the RRT algorithm. The algorithm generates random sampling points N_{rand} in space, iterates through all the nodes in the search tree and finds the nearest node $N_{nearest}$ to the sampling point N_{rand} as the expansion node of the tree. Then determine whether there is an obstacle in the line between N_{rand} and $N_{nearest}$, if there is, then discard the sample point and regenerate the sample point; if not, generate N_{new} new nodes by connecting in the direction of N_{rand} with the minimum step E as the length, and add the newly generated nodes to the random tree. The above steps are repeated until the end point is reached or the set maximum number of iterations is reached. Finally, a path from the starting point to the end point is obtained by retracing all nodes of the random tree from the end point. Since the nodes are sampled randomly, the paths obtained by this algorithm are also not globally optimal. It is the stochastic nature of the RRT path planning algorithm that causes the algorithm to have probabilistic completeness in complex environments, where the probability of the algorithm finding a feasible path converges to one given a sufficiently large number of planning iterations.

B. RRT-CONNECT ALGORITHM

Since the RRT algorithm generates sampled nodes with the same probability throughout the configuration space and does not consider expansion in the direction of the target point, spanning tree exploration is somewhat blind. To solve this problem, a bidirectional RRT algorithm, RRT-Connect with the idea of greedy extensions, was proposed [24]. Compared to the RRT algorithm, RRT-Connect makes two significant improvements: (1) It generates a random tree from the start and end state points, and the planning is completed when the two trees intersect, which greatly improves the search speed of the algorithm. (2) A node-greedy expansion strategy is used, where at each iteration of node generation, the algorithm tries to use the nearest node of another tree as the expansion direction of this tree, making the two trees intersect quickly. Also, the algorithm will continue to expand a node in that direction if it does not encounter an obstacle when expanding in that direction. If an obstacle is encountered, the algorithm will use a swap function that allows another random tree to be expanded, which largely avoids the algorithm falling into a local optimum dilemma.

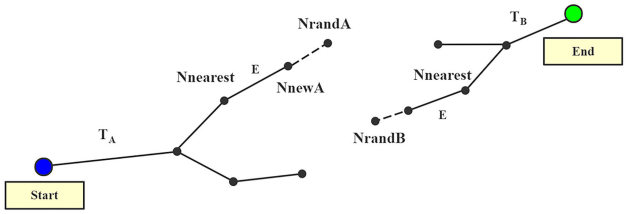

FIGURE 2. RRT-Connect algorithm spanning tree expansion process.

By using several of these strategies, RRT-Connect search speed and search efficiency have been dramatically improved. Fig. 2 shows the RRT-Connect algorithm extension process. The algorithm is initially two trees with the start and end points as root nodes, and is extended greedily, effectively reducing the extension of the random tree to unknown regions. However, RRT-Connect is similar to the RRT algorithm in that although a feasible path can be obtained after several iterations, there is no guarantee that the resulting path is optimal or sub-optimal.

C. RRT* ALGORITHM

The RRT-Connect algorithm greatly improves the speed of RRT in finding feasible paths through greedy growth and bi-directional growth strategies, but still does not consider how to optimize the feasible solution. The RRT* algorithm remedies this problem by giving the RRT algorithm the ability to be asymptotically optimal, as the number of sampling points increases, the paths obtained by the algorithm gradually converge towards the optimal path. The RRT* algorithm takes into account the cost of each node to the starting point, the lower the cost, the more likely it is to be selected as the path node. The RRT* algorithm finds the initial path in the same way as RRT, the difference being that the former does not end once it has found the initial path, but continues to generate sample points and continually updates the initial path. As the number of sample points increases, the initial path moves closer to the optimal path. Compared to the RRT and RRT-Connect algorithms, the RRT* algorithm improves the quality of the path but requires a large number of iterations and a slower convergence rate, so the planning time increases substantially.

III. IMPROVED RRT-CONNECT ALGORITHM

This section is a two-part improvement to RRT-Connect to improve the search time and search efficiency of the algorithm.

A. GENERATION OF THE THIRD NODE

Based on the idea that the RRT-Connect algorithm optimizes path search speed by constructing two random trees from the start and end points. In this paper, it is proposed to generate a third node to further optimize the iteration speed of the original RRT-Connect algorithm. In the configuration space, the starting point coordinates are initialized as $X_{start}(x_1, y_1)$

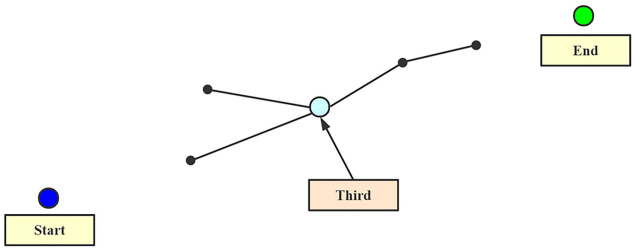

FIGURE 3. The position of the generated third node is not on an. obstacle.

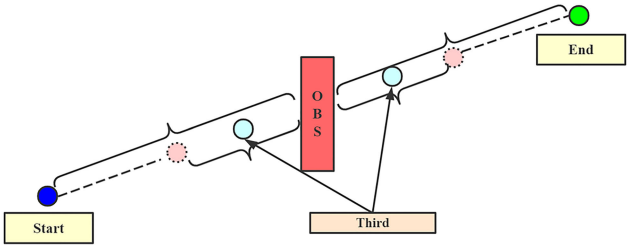

FIGURE 4. Get two candidate third nodes.

and the ending point coordinates as $X_{end}(x2, y2)$. To quickly obtain a simple and more desirable third node, we connect the start and end points into a line and take the midpoint of this line as the third node, then the coordinates of the third node $X_{mid}(x_{mid}, y_{mid})$ in the configuration space are:

$$x_{mid} = (x_1 + x_2)/2 \tag{1}$$

$$y_{mid} = (y_1 + y_2)/2 \tag{2}$$

where x_1, x_2, and y_1, y_2 are the horizontal and vertical coordinates of the start and end points in configuration space, respectively.

There are two possible scenarios for the generation of a third node in the configuration space by the above method. The first case is where the location of the generated third node does not contain an obstacle, which would indicate that only one iteration is required to find the third node, and this case is shown in Fig. 3. After generating the third node, the algorithm expands a total of four random trees including two random trees from the starting point to the third node and two random trees from the third node to the end point. The speed of searching for the proposed path planning algorithm is greatly improved.

The first case above, where the third node is found in one search is ideal. In practice, it may happen that the midpoint of the line from the start to the end is just above the obstacle. In this paper, in order to continue to ensure the efficiency of the iteration of the algorithm, we use the idea of dichotomy to continue the search for a valid third node on the basis of the third node X_{mid} generated above. The procedure is to first select the midpoint of the line that connects the starting point X_{start} with the point X_{mid} on the obstacle. This midpoint is then connected to point X_{mid} on the obstacle and the midpoint X_{mid}^{start} of this line is selected. This is used as one of the alternative valid nodes. This was done to better reflect the advantages of using a third node. The same procedure is used to select the point X end as the second alternative node on the line with X_{mid} and X_{mid}^{end} The formulae for generating these two alternative third nodes are shown in Eq. (3) and (4).

$$X_{mid}^{start}(x, y) \begin{cases} \dfrac{\frac{(x_1+x_{mid})}{2} + x_{mid}}{2} \\ \dfrac{\frac{(y_1+y_{mid})}{2} + y_{mid}}{2} \end{cases} \tag{3}$$

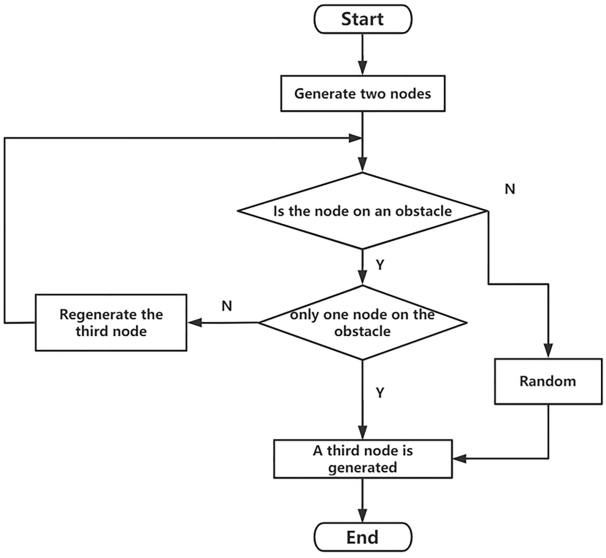

FIGURE 5. Flowchart for generating the third node.

$$X_{mid}^{end}(x, y) \begin{cases} \dfrac{\frac{(x_2+x_{mid})}{2} + x_{mid}}{2} \\ \dfrac{\frac{(y_2+y_{mid})}{2} + y_{mid}}{2} \end{cases} \tag{4}$$

where (x_1, y_1) and (x_2, y_2) are the coordinates of the start and end points in configuration space respectively. Two candidate third nodes were obtained based on this approach and the resulting nodes are shown in Fig. 4.

The two nodes obtained are then judged and if neither node is on the obstacle, then one will be chosen at random as the valid third node. If one of the two nodes is on an obstacle, the other node that is not on the obstacle is chosen as the valid third node. If both nodes generated are on the obstacle, then the two candidate third nodes will be obtained by continuing the dichotomy towards the midpoint X_{mid} based on these two alternative third nodes. Repeat the above steps until a valid third node is obtained. In short, the flow chart for the subsequent generation of the third node is shown in Fig. 5 when the third node X_{mid} generated at the very beginning is on an obstacle.

B. INCREASING THE POWER OF GUIDANCE

Generating the third node by the method in the previous subsection allows the improved RRT-Connect algorithm to improve the search speed substantially, but still does not

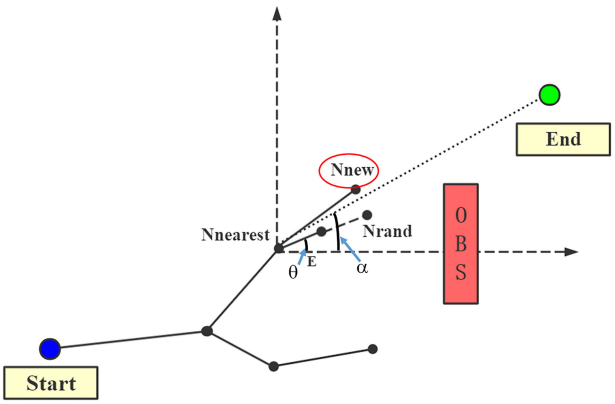

FIGURE 6. An improved process for generating new nodes.

change the problem of blind search of sampling points in the RRT-Connect algorithm path planning. To address this phenomenon, this paper proposes enhancing the search efficiency of the algorithm by reprogramming the generation of new nodes based on the bootstrap force.

We argue that the original new node N_{new} generation of the RRT-Connect algorithm is obtained by randomly sampling points and a fixed step size E, such that the bias towards the endpoint is lost. When the spanning tree is expanded, the nearest $N_{nearest}$ to the sampled point N_{rand} on the spanning tree makes an angle α with the end point X_{end} in the horizontal direction, by which the new nodes are re-planned to be more biased towards the end point. Therefore, the rules for generating the coordinates (x, y) of the new node N_{new} in this paper are shown in Eq. (5) and (6).

$$x = N_{nearest}(x) + Dcos\,\theta + kcos\,\alpha + (1-k)sin\,\alpha \quad (5)$$

$$y = N_{nearest}(y) + Dsin\,\theta + ksin\,\alpha + (1-k)cos\,\alpha \quad (6)$$

where $N_{nearest}(x)$ and $N_{nearest}(y)$ are the coordinates of the distance of the sampled point from the nearest point on the spanning tree; D is the minimum value of the distance between a fixed step E and $N_{nearest}$ and N_{rand}; θ angle is the angle enclosed by the sampling point N_{rand} and the nearest point $N_{nearest}$ on the tree in the horizontal axis direction; α angle is the angle between the end point X_{end} and $N_{nearest}$ in the direction of the horizontal axis; k is the target deviation factor, which ranges from 0 to 1. In this paper, taking 0.4 achieves the best results.

The process of generating a new node by increasing the bootstrap is shown in Fig. 6. This strategy allows the random spanning tree to consider the direction of the endpoint each time a new node is generated, so that the overall growth direction of the spanning tree in the configuration space is always biased towards the endpoint. Based on this improvement, the search efficiency of the path planning algorithm has solidly improved.

C. IMPROVED ALGORITHM IMPLEMENTATION PROCESS

In order to make the RRT-Connect path planning algorithm search faster and more efficient, we improved the entire planning process of RRT-Connect path planning by introducing the above-mentioned strategy of generating a third node and replanning for new nodes. The pseudo-code of the *ImproveExtrnd* function for random tree expansion to generate new nodes is shown in Algorithm 1.

Algorithm 1 ImproveExtend (N_{rand}, T)

Input: N_{rand}, T
Output: N_{new}

1. **if** WithoutObstacle (N_{rand}) **then**
2. $N_{nearest} \leftarrow$ Nearest (N_{rand}, T);
3. **else**
4. $N_{rand} \leftarrow$ SampleFree ();
5. Extend (N_{rand}, T);
6. D $\leftarrow$ Min (Distance (N_{rand}, $N_{nearest}$), E)
7. $N_{new}(x) \leftarrow N_{nearest}(x) + Dcos\,\theta + kcos\,\alpha + (1-k)sin\,\alpha$;
8. $N_{new}(y) \leftarrow N_{nearest}(y) + Dsin\,\theta + ksin\,\alpha + (1-k)cos\,\alpha$;
9. **if** WithoutObstacle (N_{new}) **then**
10. T.add (N_{new});
11. **else**
12. $N_{rand} \leftarrow$ SampleFree ();
13. Extend (N_{rand}, T);
14. **return** N_{new}

Among them, *WithoutObstacle* function is used to determine whether the point in the configuration space is at the obstacle; *Nearest* function aims to find the node that is closest to the sampling point N_{rand} distance in the spanning tree; *SampleFree* function generates random points in the configuration space as sampling points; *Distance* function is used to determine the Euclidean distance between two nodes of the generation.

The growth direction of the random tree is made purposeful by the method of expanding the new node in the spanning tree of Algorithm 1. Unlike the RRT-Connect algorithm, the improved algorithm in this paper generates a third node in the configuration space, so that the path of the entire algorithm is obtained by expanding it by four trees. The total path planning process of the improved algorithm is shown in the Algorithm 2 *ImproveRRT-Connect* pseudo-code.

The improved algorithm first obtains a third node through the function *SearchThirdNode*, which is implemented through Eq. (1) (2) or (3) (4). Then, four random spanning trees are initialized and the believed initialization nodes are added. T_1 and T_2 are two random trees that are oriented between the starting point and the third node, and T_3 and T_4 are two random trees that are oriented between the third node and the end point. During expansion, the *ConnectWithoutObstacle* function is used to determine if there is an obstacle between the node in the spanning tree and the newly generated node, and if so, to regenerate the new node. In this algorithm, the same greedy expansion strategy is used, i.e., if no obstacle is encountered when expanding in a certain direction, the expansion continues in that direction until an obstacle location is encountered or two trees are connected,

Algorithm 2 ImproveRRT-Connect (X_{start}, X_{end})

Input: X_{start}, X_{end}
Output: Path

1. $X_{mid} \leftarrow$ SearchThirdNode();
2. T_1.init (X_{start}), T_2.init (X_{mid}), T_3.init (X_{mid}), T_4.init (X_{end});
3. **for** $i \leftarrow 1$ to N **do**
4. $N_{rand1} \leftarrow$ SampleFree () $N_{rand2} \leftarrow$ SampleFree ();
5. $N_{nearest1} \leftarrow$ Nearest (N_{rand}, T_1) $N_{nearest2} \leftarrow$ Nearest (N_{rand}, T_4);
6. $N_{new1} \leftarrow$ ImproveExtend (N_{rand}, T_1) $N_{new2} \leftarrow$ ImproveExtend (N_{rand}, T_4)
7. **if** ConnectWithoutObstacle (N_{new1}, $N_{nearest1}$), ConnectWithoutObstacle (N_{new2}, $N_{nearest2}$), **then**
8. $T_1 \leftarrow N_{new1}$
9. $T_4 \leftarrow N_{new2}$
10. $N_{new3} \leftarrow$ ImproveExtend (N_{new1}, T_2)
11. $N_{new4} \leftarrow$ ImproveExtend (N_{new2}, T_3)
12. $T_2 \leftarrow N_{new3}$
13. $T_3 \leftarrow N_{new4}$
14. **for** $N_{new1} \neq N_{new3}$, $N_{new2} \neq N_{new4}$
15. **do** $N_{new3\text{-}temp} \leftarrow$ ImproveExtend (N_{new3}, T_2)
16. **if** ConnectWithoutObstacle ($N_{new3\text{-}temp}$, N_{new3})
17. $N_{new3} = N_{new3\text{-}temp}$
18. **do** $N_{new4\text{-}temp} \leftarrow$ ImproveExtend (N_{new4}, T_3)
19. **if** ConnectWithoutObstacle ($N_{new4\text{-}temp}$, N_{new3})
20. $N_{new4} = N_{new4\text{-}temp}$
21. **else break;**
22. **if** $N_{new1} = N_{new3}$, $N_{new2} = N_{new4}$
23. Return path (T_1, T_2) + path (T_3, T_4);
24. **else** Swap (T_1, T_2), Swap (T_1, T_2)**;**
25. **return null**;

as shown in lines 15 to 20 of the pseudo-code. To enhance the efficiency of the algorithm exploration, when two trees are expanded in opposite directions, if one of the tree obstacles meets the obstacle, a swap expansion is performed using the *Swap* function to expand the other tree, making the algorithm quickly out of the obstacle. The planning process fails when the set maximum number of iterations N is exceeded before a feasible path is found.

IV. EXPERIMENTAL SIMULATION AND ANALYSIS

In order to demonstrate the effectiveness and superiority of the improved algorithm, this paper compares, this paper compares the execution efficiency of the algorithms of RRT, RRT-Connect, RRT* and the improved RRT-Connect algorithm in simple and complex environments. The algorithm language used for the simulation is Python 3.7, the hardware platform is the Window 10 operating system, AMD Ryzen 4800U 1.8GHz CPU, and 16GB of RAM. For comparison purposes, we have named the improved algorithm IRRT-Connect.

A. SIMULATION EXPERIMENTS IN SIMPLE ENVIRONMENT

During the experiments, the corresponding parameters of the three algorithms were kept the same, with a uniform fixed step size E of 0.8. In all three environments, the experimental simulation maps ranged from (0, 50) m in the horizontal coordinates and (0, 30) m in the vertical coordinates. Fig. 7 shows the performance of the three algorithms under Environment 1. The starting point coordinates are set to (2, 2) in configuration space and the end point coordinates to (49, 24). Due to the stochastic nature of random sampling-based path planning algorithms, we conducted 50 experiments with each algorithm and the graphs show the relatively good results for each algorithm in this setting. The original RRT algorithm shown in Fig. 7 took 0.062 seconds to execute in this environment, with 405 iterations, resulting in a final path length of 65.39m. The original RRT-Connect algorithm shown in the figure has an execution time of 0.011 seconds, 83 iterations and a path length of 59.68 meters in the modified environment. The last graph shows the performance of the improved algorithm IRRT-Connect in the environment. The overall path planning time is 0.007 seconds, the number of iterations is only 15, and the path length is reduced to 54.84 meters. Table 1 shows the various average performance metrics obtained for the three algorithms after 50 experiments under environment 1. By adding a rational third node, the algorithm reduces the number of iterations by 51%, the planning time by 36% and the path length by 7% compared to the RRT-Connect algorithm before the improvement. Looking at Fig. 7 and combining the experimental data, it can be seen that the improved algorithm has a significantly lower number of iterations and a smoother path in environment 1.

In order to verify the generality of the algorithms in different environments, the performance of the algorithms in different environments and at different coordinate points is tried in this paper. The performance of the three algorithms in environment 2 is shown in Fig. 8. In this environment, the coordinates of the start and target points are (2, 26) and (47, 5) and the algorithm finds a valid path from top left to bottom right. To illustrate the effectiveness of the third node proposed in this paper, an obstacle is intentionally used in Environment 2 to be placed at the midpoint of the line connecting the start coordinates to the end coordinates, such that the first iteration of the algorithm proposed in this paper fails to find a valid third node according to equation (1) (2), forcing the algorithm to continue iterating to continue finding a valid third node according to equation (3) (4). As in Environment 1, we also conducted 50 experiments in Environment 2 and the average results obtained for the three algorithms are shown in Table 2.

As can be seen from the data in Table 2, the improved algorithm reduces the average planning time and the average number of iterations by 36.8% and 66.8% respectively compared to RRT-Connect, and the path length obtained on this basis is also reduced by approximately 4.84m. The average

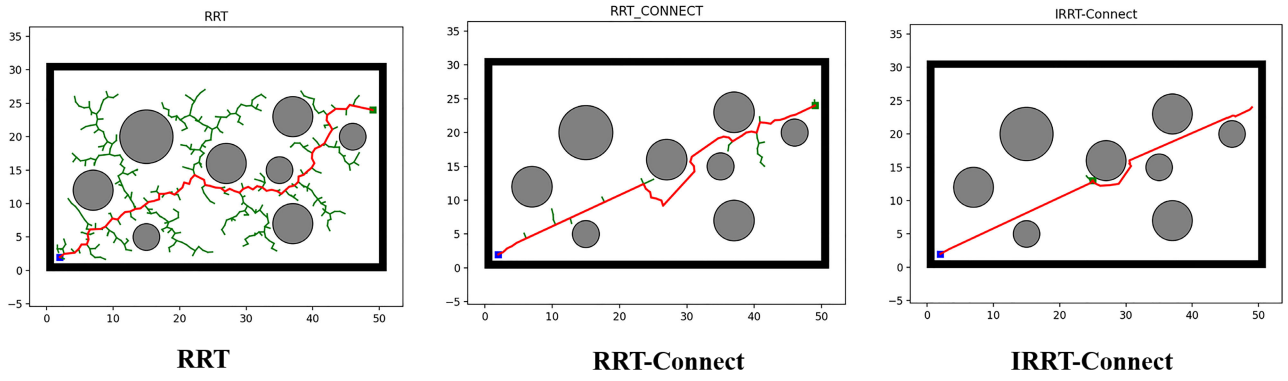

FIGURE 7. Performance of the three algorithms in Environment 1 conditions.

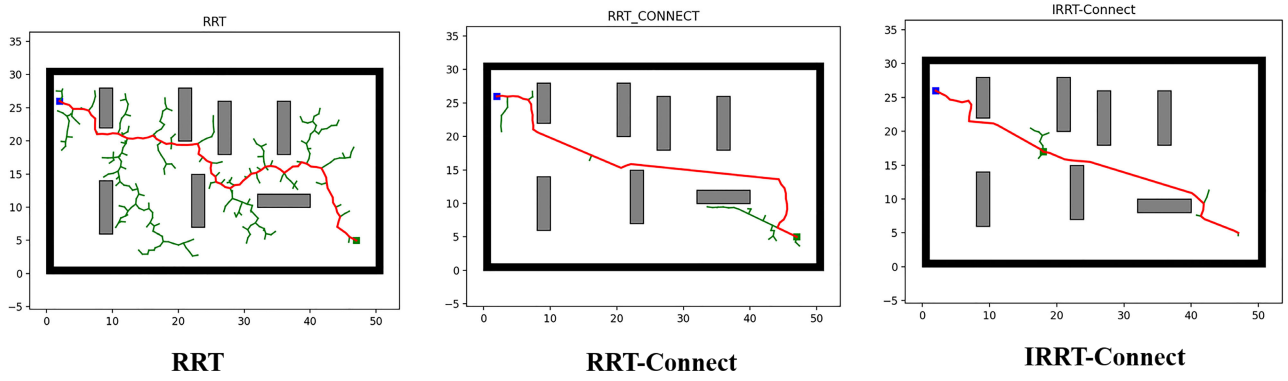

FIGURE 8. Performance of the three algorithms in Environment 2 conditions.

TABLE 1. Average results of 50 experiments with the three algorithms under environment 1.

Algorithms	Avg. Time(s)	Avg. Number of iterations(times)	Avg. Length (m)
RRT	0.061	461	64.08776827
RRT-Connect	0.011	92	59.60203369
Proposed	**0.007**	**45**	**55.67784104**

TABLE 2. Average results of 50 experiments with the three algorithms under environment 2.

Algorithms	Avg. Time(s)	Avg. Number of iterations(times)	Avg. Length (m)
RRT	0.424	594	65.40649921
RRT-Connect	0.174	190	60.33237
Proposed	**0.117**	**63**	**55.49214736**

data in Table 2 and the path planning results of the three algorithms in environment 2 in Fig. 8 show that the improved algorithm in this paper finally generates a valid third node, and the resulting path are better than the other two algorithms.

In order to verify the effectiveness of the algorithm when executed in a complex environment, Environment 3 was therefore created by combining the distribution and type of obstacles of both Environment 1 and Environment 2. In this environment, set the start and end coordinates to (2, 2) and (49, 24).

The experimental results in this environment are shown in Fig. 9. The RRT-Connect algorithm does not do a good job of smoothly transitioning along obstacles when they are encountered, and the generated paths are more convoluted after the greedy expansion of the tree swap after the spanning tree encounters an obstacle. Although both the RRT-Connect algorithm and the improved algorithm in this paper use a greedy expansion strategy, we also introduce an expansion strategy of adding new nodes with increased bootstrap, making the expansion direction more biased towards the

TABLE 3. Average results of 50 experiments with the three algorithms under environment 3.

Algorithms	Avg. Time(s)	Avg. Number of iterations(times)	Avg. Length (m)
RRT	0.454	959	69.92232093
RRT-Connect	0.142	326	67.37012684
Proposed	**0.084**	**94**	**64.41199562**

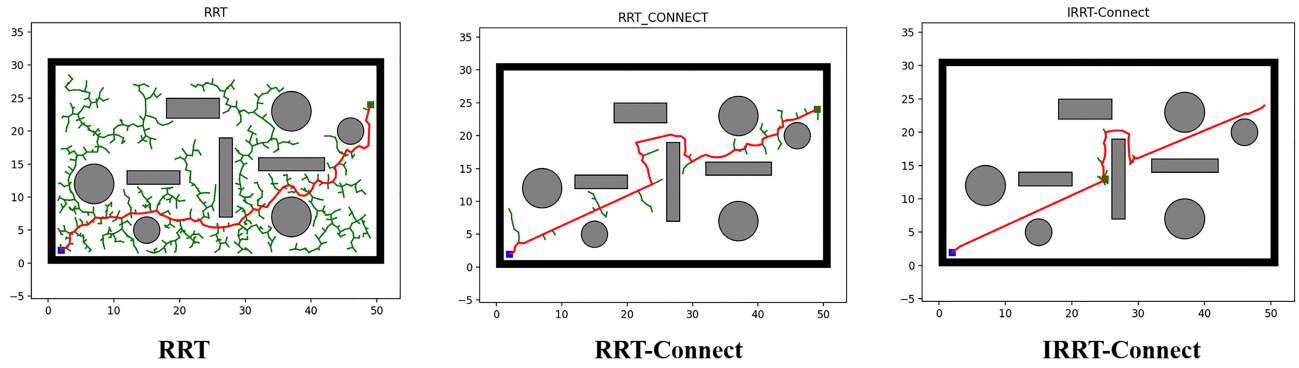

FIGURE 9. Performance of the three algorithms in Environment 3 conditions.

target point and directional. As a result, the paths obtained are straighter and a valid path can be found quickly after encountering an obstacle. The average results of the three algorithms for 50 iterations under environment 3 are shown in Table 3. The data shows that the improved algorithm proposed in this paper is able to obtain a better path in less time with fewer iterations compared to the other two algorithms.

In order to verify the stability of the algorithms, the experimental results averaged over 10, 20, 30, 40 and 50 times for each of the three algorithms in three different environments. Three performance metrics, including the number of iterations, planning time, and seek length, are compared to reflect the efficiency and quality of the algorithm's execution. As shown in Fig. 10(a)(b), the number of iterations versus pathfinding time for the three algorithms in the three environments. By generating an efficient third node, the number of iterations of the algorithm is significantly reduced for both simple and complex environmental conditions, which also allows the improved algorithm to search faster. This improvement is necessary to cope with the planning of road strength in a complex environment. Fig. 10(c) shows a comparison of the path finding lengths of the three algorithms in the three environments. The improved algorithm finds a shorter path length than the other two algorithms, both after a small number of experiments (10 times) and a large number of experiments (50 times). It is possible that this is due to the fact that we have reprogrammed the spanning tree in expanding the new node method so that each iteration generates a new node in the direction of the target, increasing the search speed of the algorithm. Combined with Fig. 10 it can be illustrated that the improved path planning algorithm searches faster and explores better in multiple environments.

B. SIMULATION EXPERIMENTS IN COMPLEX ENVIRONMENTS

In order to compare the execution performance of the improved algorithm in complex environments, three more complex environments including a dense obstacle environment, a U-shaped environment and a narrow lane environment were created for experiments. In this experiment, the map environment was expanded by enlarging a raster with a map size of 50 $\times$ 30 in the simple environment to 120 $\times$ 100 for the experiment. Furthermore, to demonstrate the superiority of the proposed algorithm in finding paths, the asymptotically optimal RRT* algorithm is introduced under the condition of a complex environment. The algorithm improves parent node selection by using a cost function to select the node with the smallest cost in the domain of the expanded node as the parent, while the nodes in the existing tree are reconnected after each iteration, thus ensuring the computational complexity and asymptotically optimal solution. Environment 4 is a complex environment under a large map containing more dense circular obstacles and rectangular obstacles. In this environment, the start and end coordinates are set to (2, 2) and (110, 98) respectively, and the generated paths and results of the four algorithms are shown in Fig. 11. The results of this run are written after the serial number of each algorithm, where t represents time, d represents the path length and i represents the number of iterations. The results in the figure show that although the RRT-Connect algorithm uses a greedy search strategy and has a partially straighter route, there are still cases where blind exploration causes the path to twist and turn. For the RRT* algorithm, the maximum number of iterations set in this paper is 10,000. After 10,000 iterations, the algorithm has a good improvement over the RRT and RRT-Connect path lengths, but the execution time

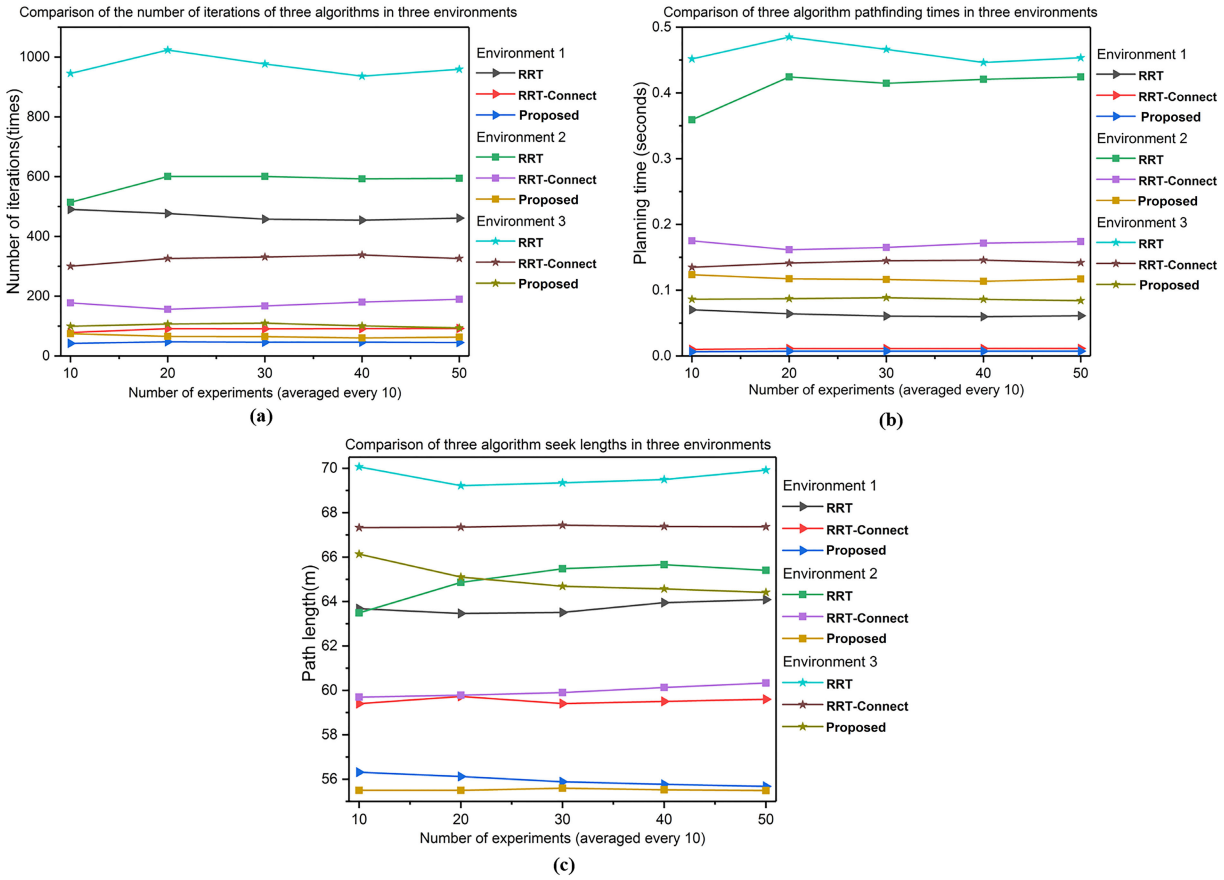

FIGURE 10. Performance of the three algorithms under three environmental conditions.

reaches 251.5 seconds because the algorithm needs to gradually optimize the paths. Fig. 11(d) shows the results of the improved algorithm proposed in this paper executed under environment 4. Due to the dense obstacles, a valid third node was not generated for the first time, and a valid third node was generated after another search. The improved algorithm has a straighter path in this environment, resulting in a path length of only 164.85m, and takes less time and fewer iterations.

Environment 5 is another complex environment, a U-shaped environment. In the U-space state, the algorithm spanning tree has only one exit and is therefore more challenging. In this environment, the coordinates of the starting point are set to be in U-space at (58, 40), and the coordinates of the ending point are on the other side of the U-shaped region at (58, 85). The results of the execution of the four algorithms are shown in Fig 12. Because the random search tree needs to escape the U-shaped region from the only exit, the number of iterations for all four algorithms in this setting has been increased relative to the previous setting. For the IRRT-Connect algorithm proposed in this paper, since the third node generation will still fall within the U-shaped region, we eliminate the generation of the third node in the sub environment and only run the algorithm with increased bootstrap power. The results of the algorithm execution in Fig 12(d) show that the improved algorithm escapes from the

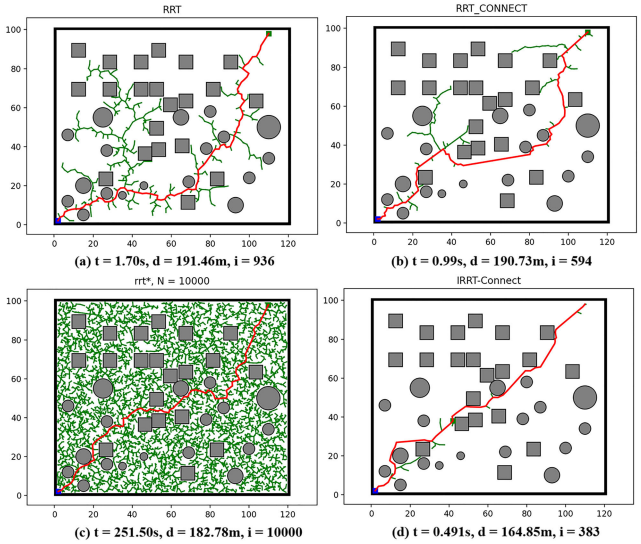

FIGURE 11. Performance of the four algorithms in Environment 4 conditions.

U-shaped region and then approaches the end point with a smoother trend. The reason for this may be that the algorithm makes the algorithm search more purposeful by adding bootstrap force to the newly generated nodes based on the greedy search strategy. In the sub-context, there is a substantial

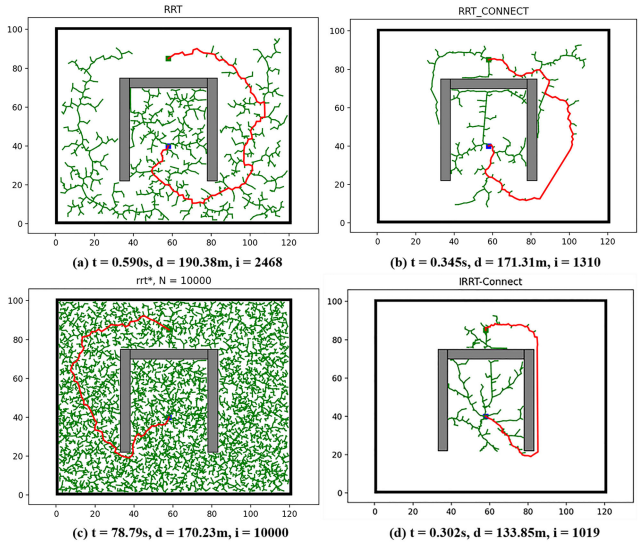

FIGURE 12. Performance of the four algorithms in Environment 5 conditions.

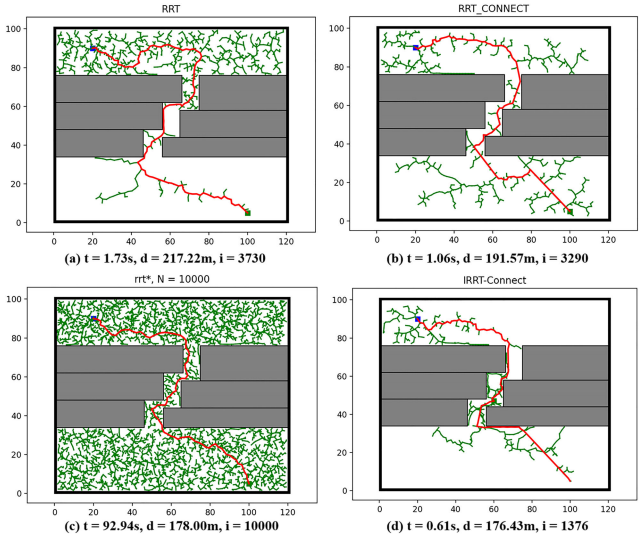

FIGURE 13. Performance of the four algorithms in Environment 6 conditions.

improvement in path length and search time compared to the asymptotically optimal RRT* algorithm.

Environment 6 is a narrow lane environment that examines the algorithm's ability to search in a narrow environment. In this environment, the starting and ending coordinates are set to (20, 90) and (100, 5) respectively, and the algorithm needs to pass through the only narrow path from top to bottom to reach the target point, and the experimental results are shown in Fig 13. Similar to the previous environments, the RRT algorithm requires a high number of iterations to find a feasible path and suffers from a severe case of blind search over narrow lanes. The RRT* algorithm achieved a path length of 178 meters after 92.95 seconds in this environment. The algorithm proposed in this paper reduces the number of iterations and shortens the path planning time compared to the well-performing RRT* algorithm in a narrow channel environment, although the difference in path length is not significant, which facilitates a fast and efficient path planning in a real environment.

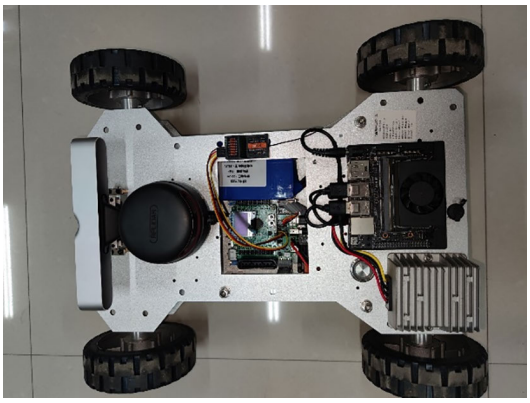

FIGURE 14. The mobile robot used in this paper.

To better evaluate the performance of the algorithms, similar to the experiments in the simple environment, 50 experiments were conducted in this paper in three more complex environments. The experiments compared the performance metrics of the four algorithms in the three complex environments in terms of the number of iterations, planning time and planning distance, and the experimental results are shown in Table 4. Based on the statistical results, it can be seen that the improved algorithm proposed in this paper achieves good convergence compared to the other three algorithms under the conditions of complex environments, both in terms of the number of iterations and planning time, which are reduced to a low level. The RRT* algorithm is set for 10,000 iterations at a time, and as the number of sample points increases, the algorithm gradually moves closer to the asymptotic optimum. In particular, in the case of environment 6, the difference in path length between the RRT* algorithm and the algorithm proposed in this paper is not significant. However, the proposed algorithm uses a much smaller number of iterations and time elapsed, which may be explained by the fact that the proposed algorithm makes the spanning tree converge faster by the strategy of increasing the bootstrap and using the third node to speed up the algorithm search.

C. IMPLEMENTATION ON ROS MOBILE ROBOTS

In order to test the ability of the algorithm proposed in this paper for path planning in real-world application scenarios, the algorithm was ported to the Ros mobile robotics platform based on Ubuntu 18.04. The robot used in the experiments is a four-wheeled cart with a Silan A2 LIDAR for modeling the environment and an Nvidia Xavier NX as the main control with the ROS operating system and an STM32 for ground control and signal transmission, the mobile robot is shown in Fig 14.

The robot localization and mapping process relies on simultaneous localization and mapping (SLAM) algorithms to model the environment [30]. At present, the mainstream mapping algorithm is based on the LIDAR sensor algorithm, which includes several steps: sensor information reading,

TABLE 4. Statistical results of 50 experiments with four algorithms in complex environments.

Test environment	Algorithm	i_{min}	i_{max}	i_{avg}	t_{min}	t_{max}	t_{avg}	d_{min}	d_{max}	d_{avg}
Env 4	RRT	921	1013	957	1.70	2.16	1.89	190.76	200.1	194.32
	RRT-Connect	562	612	593	0.89	1.34	1.01	186.32	192.52	190.34
	RRT*	10000	10000	10000	248.23	262.78	253.34	180.46	184.55	182.63
	Proposed	376	412	386	0.47	0.53	0.49	162.34	171.15	164.96
Env 5	RRT	2342	2661	2572	0.56	0.78	0.63	188.65	194.43	189.78
	RRT-Connect	1255	1460	1302	0.31	0.42	0.33	169.43	181.11	171.22
	RRT*	10000	10000	10000	77.15	78.47	79.16	169.45	175.69	170.18
	Proposed	979	1142	1017	0.29	0.35	0.30	131.39	140.37	134.87
Env 6	RRT	3545	3948	3769	1.68	1.86	1.74	216.29	220.16	218.25
	RRT-Connect	3145	3538	3288	0.96	1.13	1.09	190.98	193.44	192.37
	RRT*	10000	10000	10000	91.48	96.37	93.10	177.34	181.26	179.42
	Proposed	1247	1469	1382	0.58	0.66	0.63	175.32	178.10	176.88

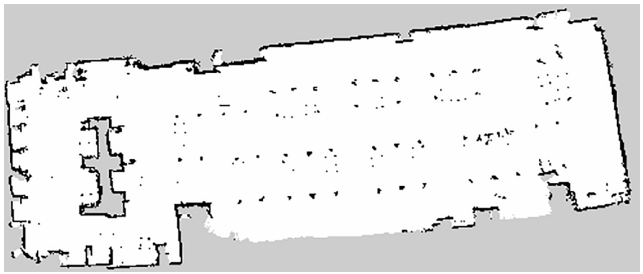

FIGURE 15. Modelling diagrams in real environments.

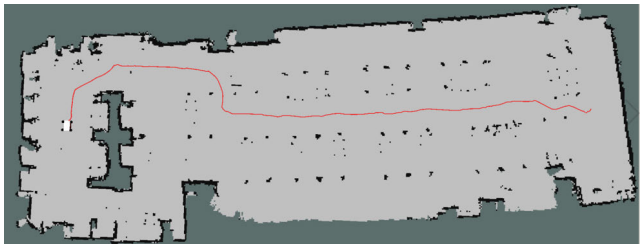

FIGURE 16. Execution results of the trolley in a real environment.

front-end data fusion, back-end positional and map optimization, loopback detection and mapping. This paper uses the filter-based particle filter Gmapping algorithm [31]. The Gmapping algorithm separates the localization and mapping processes, with localization being carried out first and then mapping. The results of the environmental modelling of the laboratory are shown in Fig 15, where the white areas are explored locations, the black dotted blocks are obstacle locations and the grey blocks are unsearched areas.

In the ROS operating system, the improved algorithm needs to be written to the *move_base* module in the ROS navigation framework and subsequently route planning is performed. The inputs to the navigation system include LiDAR information, odometer information, map information and robot position information. Based on the laboratory modeling map completed using the Gmapping algorithm in the previous vignette, the start and end points were selected on the map and the algorithm planned the completed path as shown in Fig 16. As can be seen from the figure, the algorithm generates a shorter feasible path through the environment and is able to smoothly transition past obstacles when encountered, effectively overcoming the algorithm's tendency to fall into local optima under large maps, resulting in smoother robot motion and validating the effectiveness of the proposed algorithm working on actual robots.

V. CONCLUSION

Path planning for mobile robots in complex environments is one of the key problems to be solved in the research field. With its simplicity and fast search capability, the sampling-based path planning algorithm is adept at solving planning problems in multidimensional spaces and complex environments. This paper is based on the RRT-Connect algorithm. Firstly, in order to further accelerate the efficiency of the algorithm in space exploration, the algorithm is expanded in the form of a quadtree by generating a third node. Then, in order to solve the blind search problem of RRT-Connect, the generation of new nodes is reprogrammed by increasing the bootstrap, so that the new nodes are expanded in favor of the target point each time, which speeds up the search efficiency of the improved algorithm.

In order to verify the effectiveness and generality of the proposed algorithm, six environment maps with different complexities were constructed and averaged for performance metrics after 50 experiments. For the simpler environment 1, the improved algorithm reduces the number of iterations by 91% and 51%, the pathfinding time by 88% and 36%, and the resulting path length by 13% and 7% compared to the RRT and RRT-Connect algorithms. At the same time, three more complex environments have been created, namely a densely distributed obstacle environment, a U-shaped environment and a narrow lane environment. The performance of four algorithms executed in a complex environment was statistically compared. The experimental results show that the improved algorithm can show excellent performance in several aspects in a complex environment, which may be attributed to the increased bias of the improved algorithm, which improves the exploration efficiency and proves the generality of the algorithm. Finally, the proposed path planning algorithm is ported to the ROS robot to verify the effectiveness of the algorithm in a real-world environment.

The path planning algorithm studied in this paper was carried out in a two-dimensional space after modeling the environment, and none of the obstacle points set were considered in terms of height. However, in real scenarios, there may be parts of the area where the roads are uneven and obstacles are irregular and three-dimensional in their spatial state, so real-time path planning in a three-dimensional environment is a further direction for future research.

ACKNOWLEDGMENT

The authors are grateful for the experimental equipment and platform provided by the Machine Intelligence and Vision Laboratory, Zhejiang University of Science and Technology, Zhejiang.

REFERENCES

[1] X. Gao, J. Li, L. Fan, Q. Zhou, K. Yin, J. Wang, C. Song, L. Huang, and Z. Wang, "Review of wheeled mobile robots' navigation problems and application prospects in agriculture," *IEEE Access*, vol. 6, pp. 49248–49268, 2018.
[2] F. Weidinger, N. Boysen, and D. Briskorn, "Storage assignment with rack-moving mobile robots in KIVA warehouses," *Transp. Sci.*, vol. 52, no. 6, pp. 1479–1495, Dec. 2018.
[3] T. Namba and Y. Yamada, "Risks of deep reinforcement learning applied to fall prevention assist by autonomous mobile robots in the hospital," *Big Data Cognit. Comput.*, vol. 2, no. 2, p. 13, Jun. 2018.
[4] F. H. Ajeil, I. K. Ibraheem, M. A. Sahib, and A. J. Humaidi, "Multi-objective path planning of an autonomous mobile robot using hybrid PSO-MFB optimization algorithm," *Appl. Soft Comput.*, vol. 89, Apr. 2020, Art. no. 106076.
[5] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Math.*, vol. 1, no. 1, pp. 269–271, Dec. 1959.
[6] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. Syst. Sci. Cybern.*, vol. SSC-4, no. 37, pp. 28–29, Jul. 1972.
[7] O. Khatib, "Real-time obstacle avoidance system for manipulators and mobile robots," *Int. J. Robot. Res.*, vol. 5, no. 1, pp. 90–98, 1986.
[8] M. Dorigo and V. Maniezzo, "Ant system: Optimization by a colony of cooperating agents," *IEEE Trans. Syst., Man, Cybern., B, Cybern.*, vol. 26, no. 1, pp. 29–41, Feb. 1996.
[9] H.-M. Zhang, M.-L. Li, and L. Yang, "Safe path planning of mobile robot based on improved A* algorithm in complex terrains," *Algorithms*, vol. 11, no. 4, p. 44, Apr. 2018.
[10] W. Gao, Q. Tang, B. Ye, Y. Yang, and J. Yao, "An enhanced heuristic ant colony optimization for mobile robot path planning," *Soft Comput.*, vol. 24, pp. 1–12, Nov. 2020.
[11] M. Nazarahari, E. Khanmirza, and S. Doostie, "Multi-objective multi-robot path planning in continuous environment using an enhanced genetic algorithm," *Expert Syst. Appl.*, vol. 115, pp. 106–120, Jan. 2019.
[12] U. Orozco-Rosas, O. Montiel, and R. Sepúlveda, "Mobile robot path planning using membrane evolutionary artificial potential field," *Appl. Soft Comput.*, vol. 77, pp. 236–251, Apr. 2019.
[13] U. Orozco-Rosas, K. Picos, and O. Montiel, "Hybrid path planning algorithm based on membrane pseudo-bacterial potential field for autonomous mobile robots," *IEEE Access*, vol. 7, pp. 156787–156803, 2019.
[14] A. Maoudj and A. Hentout, "Optimal path planning approach based on Q-learning algorithm for mobile robots," *Appl. Soft Comput.*, vol. 97, Dec. 2020, Art. no. 106796.
[15] A. G. D. S. S. Junior, D. H. D. Santos, A. P. F. D. Negreiros, J. M. V. B. D. S. Silva, and L. M. G. Gonçalves, "High-level path planning for an autonomous sailboat robot using Q-Learning," *Sensors*, vol. 20, no. 6, p. 1550, Mar. 2020.
[16] M. Zhao, H. Lu, S. Yang, and F. Guo, "The experience-memory Q-learning algorithm for robot path planning in unknown environment," *IEEE Access*, vol. 8, pp. 47824–47844, 2020.
[17] S. LaValle. (1998). *Rapidly-Exploring Random Trees: A New Tool for Path Planning*. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.35.1853&rep=rep1&type=pdf
[18] H. Zhang, Y. Wang, J. Zheng, and J. Yu, "Path planning of industrial robot based on improved RRT algorithm in complex environments," *IEEE Access*, vol. 6, pp. 53296–53306, 2018.
[19] K. Wei and B. Ren, "A method on dynamic path planning for robotic manipulator autonomous obstacle avoidance based on an improved RRT algorithm," *Sensors*, vol. 18, no. 2, p. 571, 2018.
[20] J. Xu and K.-S. Park, "A real-time path planning algorithm for cable-driven parallel robots in dynamic environment based on artificial potential guided RRT," *Microsyst. Technol.*, vol. 26, no. 11, pp. 3533–3546, Nov. 2020.
[21] F. Kiani, A. Seyyedabbasi, R. Aliyev, M. U. Gulle, H. Basyildiz, and M. A. Shah, "Adapted-RRT: Novel hybrid method to solve three-dimensional path planning problem using sampling and metaheuristic-based algorithms," *Neural Comput. Appl.*, vol. 33, pp. 1–31, Jun. 2021.
[22] G. P. Kontoudis and K. G. Vamvoudakis, "Kinodynamic motion planning with continuous-time Q-learning: An online, model-free, and safe navigation framework," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 12, pp. 3803–3817, Dec. 2019.
[23] Z. Liu, F. Lan, and H. Yang, "Partition heuristic RRT algorithm of path planning based on Q-learning," in *Proc. IEEE 4th Adv. Inf. Technol., Electron. Automat. Control Conf. (IAEAC)*, Dec. 2019, pp. 386–392.
[24] J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *Proc. ICRA. Millennium Conf. IEEE Int. Conf. Robot. Automat. Symposia*, San Francisco, CA, USA, Apr. 2000, pp. 995–1001.
[25] J.-G. Kang, D.-W. Lim, Y.-S. Choi, W.-J. Jang, and J.-W. Jung, "Improved RRT-connect algorithm based on triangular inequality for robot path planning," *Sensors*, vol. 21, no. 2, p. 333, Jan. 2021.
[26] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. J. Robot. Res.*, vol. 30, no. 7, pp. 846–894, 2011.
[27] J. Qi, H. Yang, and H. Sun, "MOD-RRT*: A sampling-based algorithm for robot path planning in dynamic environment," *IEEE Trans. Ind. Electron.*, vol. 68, no. 8, pp. 7244–7251, Aug. 2021.
[28] I. Noreen, A. Khan, H. Ryu, N. L. Doh, and Z. Habib, "Optimal path planning in cluttered environment using RRT*-AB," *Intell. Service Robot.*, vol. 11, no. 1, pp. 41–52, Jan. 2018.
[29] Lozano-Perez, "Spatial planning: A configuration space approach," *IEEE Trans. Comput.*, vol. C-32, no. 2, pp. 108–120, Feb. 1983, doi: 10.1109/TC.1983.1676196.
[30] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós, "ORB-SLAM: A versatile and accurate monocular SLAM system," *IEEE Trans. Robot.*, vol. 31, no. 5, pp. 1147–1163, Oct. 2015.
[31] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with rao-blackwellized particle filters," *IEEE Trans. Robot.*, vol. 23, no. 1, pp. 34–46, Feb. 2007.

JIAGUI CHEN received the B.S. degree from the School of Computer and Engineering, Jiangsu University, Zhenjiang, China, in 2019. He is currently pursuing the M.S. degree with the School of Information and Electronic Engineering, Zhejiang University of Science and Technology, Zhejiang. His research interests include computer vision and path planning.

YUN ZHAO received the Postdoctoral (Ph.D.) degree in engineering from Zhejiang University. She is currently a Professor with the School of Information and Electronic Engineering, Zhejiang University of Science and Technology. Her research interests include artificial intelligence and pattern recognition.

XING XU received the Ph.D. degree in engineering from the Zhejiang University of Technology. He is currently a Professor with the School of Mechanical and Energy Engineering, Zhejiang University of Science and Technology. His research interests include mobile robotics and intelligent transportation.