# Implementation of Improved RRT-Connect and Comparison with the Conventional RRT-Connect Algorithm

Mothish Raj Venkatesan Kumararaj
Satish Vennapu
*Engineering Department, University of Maryland, College Park, Prince George County, mr2997@umd.edu, satish@umd.edu*

**Abstract:** Path planning is a crucial component in mobile robotics, enabling intelligent navigation. However, traditional path planning methods often struggle with complex deterministic environment modeling, leading to local minima. In contrast, sampling-based algorithms, such as Rapidly Exploring Random Trees (RRT-Connect), offer faster feasibility checking. In this paper, we refer to the **Improved RRT-Connect-Based Path Planning Algorithm for Mobile Robots** by JIAGUI CHEN, YUN ZHAO1, AND XING XU2 where the authors address the two issues with the traditional RRT-Connect algorithms. They are the exploration speed of the spanning trees and the blind search feature of the conventional RRT-Connect algorithm. Addressing these issues, they propose the improved RRT-Connect algorithm (IRRT-Connect). The IRRT-Connect algorithm introduces a third node in the configuration space to accelerate the search process, utilizing a quadruple tree for greedy expansion. Furthermore, they propose a guidance mechanism that biases the expansion towards the target point, enhancing exploration efficiency. We are re-creating the improvements suggested by the authors of the paper and comparing the variants individually, to validate the proposed algorithms, we compare the performance of three algorithms in the same environment through simulation experiments.

***Key Words: RRT, RRT*, RRT Connect, Improved RRT Connect (IRRT)***

## 1 Introduction

Mobile robots have been extensively used in several fields recently, including industry, healthcare, agriculture, and services, thanks to the constant advancement of hardware equipment. For mobile robots, path planning is both a crucial technology and a big challenge. Robots use path planning to find the optimal or sub-optimal route from a given start point to a specified goal point based on various evaluation criteria (such as planning time, path length, etc.). Depending on whether or not environmental information is known, planning tasks can be divided into two broad categories. They are global and local path planning. To solve the path planning problem, several algorithms have been devised. The Dijkstra method is one solution to the planning problem founded on the notion of greed and there are heuristics based namely the A* algorithm, the artificial potential field algorithm, the bionic ant colony algorithm, etc. Some researchers modified these fundamental algorithms to fit their needs and tasks. Zhang et al, Gao et al and Nazarahari et al generated a safer path, augmented the heuristic, and combined the path length into a multi-objective path planning for mobile robots in a complex environment respectively. The aforementioned conventional path planning algorithms exhibit an exponential rise in computational complexity in complex situations and have a propensity towards reaching the local minimum. The completion criteria of the algorithms must be lowered to increase their efficiency and practicality. The computational complexity of the technique is reduced by the probabilistic completeness and resolution completeness of random sampling-based algorithms, which are frequently used planning algorithms. Among the sampling-based path planning algorithms, the most prominent one is the Rapidly Exploring Random Trees (RRT) algorithm and its improved versions mainly RRT* and RRT-Connect. The RRT algorithm is probabilistically complete and samples uniformly and randomly in the configuration space. Several improvements were introduced to address the speed and efficiency of the basic RRT algorithm such as adaptive node expansion strategy, directional node expansion, the artificial field algorithm, Grey Wolf Optimization, and the Q-Learning reinforcement learning. In order to increase the search speed of the random spanning trees, LaValle proposed the RRT-Connect to explore directionally, generating two random trees from the start node and the goal node. Kang et al proposed an RRT-Connect algorithm based on triangular inequality to reduce the path length. In this paper, the authors' contributions include the following:

1. To address the increase in the speed of the exploration of random spanning trees, a simple and efficient third node is generated based on the concept of dichotomous points to produce four spanning trees.
2. To address the blind search feature of the conventional RRT-Connect algorithm, the guiding force increased towards the direction of the target point when expanding the new nodes.

## 2 Related Works

### 2.1 RRT

The Rapidly Exploring Random Trees (RRT) algorithm is a sampling-based motion planning algorithm that efficiently explores and constructs a collision-free path for a robot or other autonomous system in complex, high-dimensional environments. The RRT algorithm starts with an initial configuration and iteratively grows a tree by randomly sampling points in the configuration space and connecting them to the nearest point in the existing spanning tree. This process creates a tree-like structure with branches that expand toward unexplored regions of the environment. The algorithm continues until the goal configuration is reached or a maximum number of iterations is reached. The different steps involved in the RRT algorithm are as follows:

1. Initialize the tree with a single node at the start configuration. This is the root node of the tree.
2. Repeat the following steps until a path is generated or a maximum number of iterations is reached:
    (a) Sample a random point N$_{random}$ in the configuration space that can be done randomly or with a bias toward the goal configuration.
    (b) Find the nearest node N$_{nearest}$ in the tree to the sampled point N$_{rand}$. This is typically done by measuring the Euclidean distance between the sampled point and all nodes in the tree and selecting the closest one.
    (c) Generate a new node N$_{new}$ by extending the tree towards the sampled point with a certain step size. This is done by moving from the nearest node toward the sampled point with a certain distance, usually a fixed step size.
    (d) If the new node does not collide with obstacles in the environment, add it to the tree. This involves checking for collisions between the new node and the obstacles in the environment. If the new node is collision-free, it is added to the tree as a new leaf node.
3. If a path is found, return it. Otherwise, return failure. A path is found if there exists a leaf node in the tree that is within a certain distance of the goal configuration. If a path is found, it can be reconstructed by following the path from the goal configuration back to the start configuration along the edges of the tree. If no path is found, return failure.

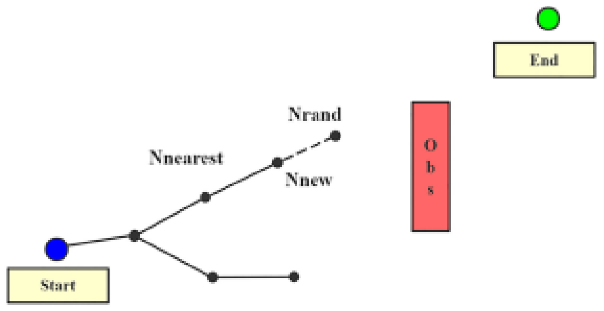

FIGURE 1. RRT

## 2.2 RRT*

The RRT* algorithm builds upon the basic RRT framework by incorporating a cost-to-come function that assigns a cost to each node in the tree, reflecting the cost of reaching that node from the initial configuration. The algorithm also maintains a set of nearby nodes for each newly added node, instead of connecting only to the nearest node as in the basic RRT algorithm. This allows for more efficient re-wiring of the tree, as nearby nodes can be updated with improved paths if a better path is found.

## 2.3 RRT Connect

In RRT-Connect, two separate trees are grown simultaneously, starting from the initial configuration (start node) and the goal configuration (goal node), respectively. These trees are expanded by randomly sampling points in the configuration space, and new points are connected to the nearest points in their respective trees. The trees are then connected by attempting to connect a new point from one tree to the other, forming a potential path between the start and goal configurations. If the path is valid, meaning it does not collide with obstacles in the environment, it is considered a potential solution path. In case the algorithm comes across an obstacle, it will utilize a swapping function that enables the expansion of another random tree. This approach greatly reduces the likelihood of the algorithm getting stuck in a local optimum situation. By using these strategies, the RRT-Connect search speed and search efficiency are increased drastically. In the beginning, the algorithm consists of two trees, each with the starting and ending points as the initial nodes. The trees are then grown in a manner that prioritizes proximity to the target, which helps to limit exploration of uncharted 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.

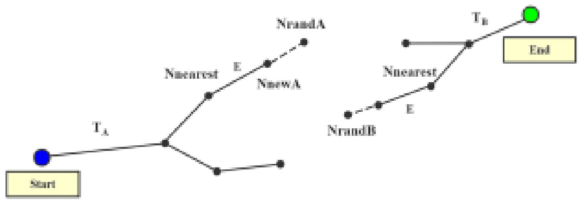

FIGURE 2. RRT Connect

## 2.4 Improved RRT-Connect

IRRT connect is based on the RRT connect algorithm. IRRT Connect is a two-part improvement to RRT-Connect to improve the search time and search efficiency of the algorithm.

1. Generation of the third node at the mid point.
2. Increasing the power of guidance.

## 3 Methodology

### 3.1 Generation of the third node at the mid point.

To quickly obtain a desirable third node, we connected the start and end points into a line and take the midpoint of this line as the third node which is given by the formula:

$$x_{mid} = (x_1 + x_2)/2 y_{mid} = (y_1 + y_2)/2 \qquad (1)$$

Where start point X$_{start}$ and end point X$_{end}$ coordinates are (x$_1$,y$_1$) and (x$_2$,y$_2$)

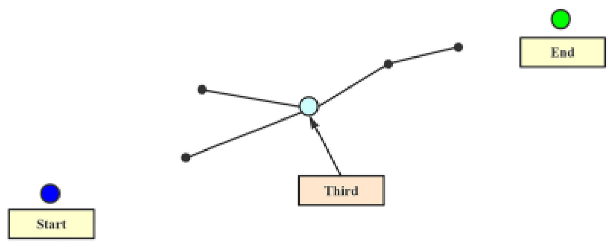

FIGURE 3. IRRT Connect with third node in mid point

There are two scenarios in which a third node can be generated in the configuration space using the method described above. In the first case, the location of the generated third node is obstacle-free, requiring only one iteration to find the third node. Once the third node is generated, the algorithm expands a total of four random trees: one tree from the third node to the start node, one from the third node to the goal node, one from the start node, and another from the goal node. In the second case, the midpoint of the line from the start to the end is just above the obstacle. This paper proposed a method to maintain the algorithm's efficiency by utilizing the concept of dichotomy to further search for a valid third node based on the previously generated third node X$_{mid}$. The approach involves selecting the midpoint of the line connecting the starting point X$_{start}$ with the point X$_{mid}$ on the obstacle and using it as a starting point for the next iteration. This midpoint is then connected to a point X$_{mid}$ on the obstacle and the midpoint X$_{mid}^{start}$ of this line is selected. The same procedure is used to select the point X end as the second alternative node on the line with X$_{mid}$ and X$_{mid}^{start}$. The formulae are as follows:

$$X_{mid}^{start}(x,y) \begin{cases} \dfrac{\frac{(x_1+x_{mid})}{2} + x_{mid}}{2} \\ \dfrac{\frac{(y_1+y_{mid})}{2} + y_{mid}}{2} \end{cases} \qquad X_{mid}^{end}(x,y) \begin{cases} \dfrac{\frac{(x_2+x_{mid})}{2} + x_{mid}}{2} \\ \dfrac{\frac{(y_2+y_{mid})}{2} + y_{mid}}{2} \end{cases}$$

FIGURE 4. Mid point formula

After obtaining two nodes, they are evaluated to check if they intersect with any obstacle in the environment. If neither node intersects with an obstacle, one of them is randomly selected as the third node. However, if one of the two nodes intersects with an obstacle, the other node that doesn't intersect with any obstacle is chosen as the third node. In case both nodes are on an obstacle, a dichotomy is continued towards the midpoint X$_{mid}$ based on the two alternative third nodes. These steps are repeated until a valid third node is obtained.

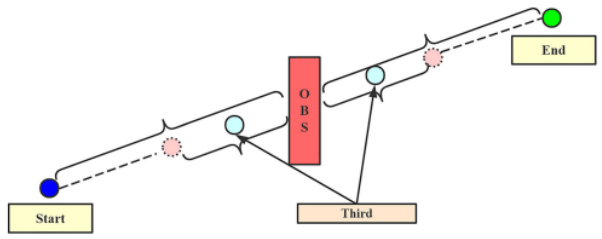

FIGURE 5. IRRT with the third node in obstacle scenario

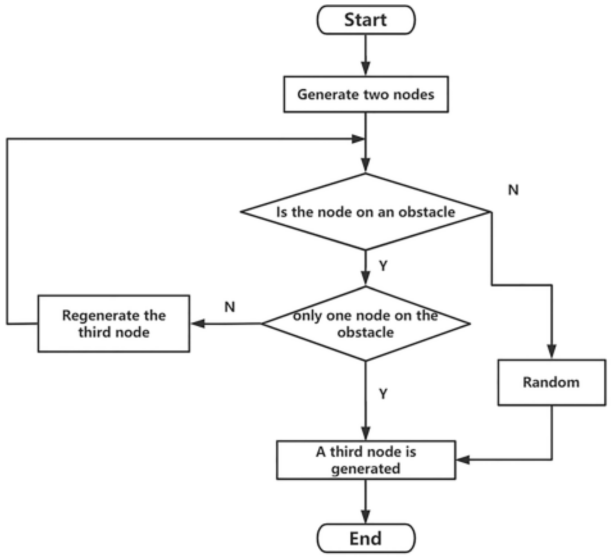
Start
Generate two nodes
Is the node on an obstacle
N
Y
Regenerate the third node
N
only one node on the obstacle
Y
Random
A third node is generated
End

FIGURE 6. Algorithm for midpoint IRRT

For our case, we considered the first scenario where there is no obstacle at the area of the third node since the path is very complicated as the node increases and will not be effective. The performance has been compensated with the guidance improvement mentioned below.

### 3.2 Increasing the power of guidance

The search efficiency of the algorithm is improved by reprogramming the generation of new nodes based on the bootstrap force. The authors claim that the initial generation of a new node N$_{new}$ in the RRT-Connect algorithm involves random point sampling and a fixed step size E, which leads to a loss of bias towards the endpoint. As the tree is expanded, the nearest node N$_{nearest}$ to the sampled point N$_{rand}$ on the tree forms an angle with the endpoint $X_{end}$ in the horizontal direction. By re-planning the new nodes with this angle, they can be more biased towards the endpoint.

The rules for generating the coordinates (x, y) of the new node $N_{new}$ in this paper are:

$$x_{new} = N_{nearest}(x) + Dcos(\theta) + Kcos(\alpha) + (1-k)sin(\alpha) \quad (2)$$

$$y_{new} = N_{nearest}(y) + Dcos(\theta) + Ksin(\alpha) + (1-k)cos(\alpha) \quad (3)$$

where the coordinates of the distance between the sampled point and the nearest point on the spanning tree are represented by $N_{nearest}$(x) and $N_{nearest}$(y), respectively. The distance between a fixed step E and $N_{nearest}$ as well as $N_{rand}$ is measured and denoted as D. The angle formed by the sampled point $N_{rand}$ and the nearest point $N_{nearest}$ on the tree in the horizontal axis direction is $\theta$, while the angle between the endpoint $X_{nend}$ and $N_{nearest}$ in the direction of the horizontal axis is $\alpha$. The target deviation factor k is taken as 0.4.

```
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 $N_{new}$
```

FIGURE 7. Algorithm for random tree expansion

In our implementation of the guidance improvement, we did not consider the third node and the results were satisfactory even without the generation of the third node. From the paper, we have taken the two improvements and implemented them separately instead of combining them together. A minor improvement was made from the above logic to handle the trap in the path generation, we are alternating between the traditional action set for generating the random node and the improved action set for the guidance when the generated path gets stuck in one position.

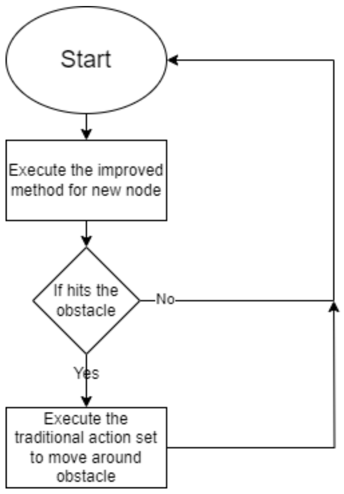
Start
Execute the improved method for new node
If hits the obstacle
No
Yes
Execute the traditional action set to move around obstacle

FIGURE 8. Algorithm for switching between the action set

## 4 Results

As random sampling-based path planning algorithms are inherently probabilistic, we performed 50 trials for each algorithm, and the charts demonstrate that each algorithm performed relatively well under these conditions. During the experiments, the corresponding parameters of the three algorithms were kept the same, with a uniform fixed step size E in all circumstances.

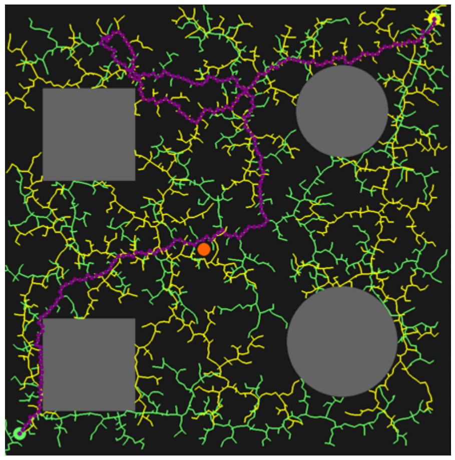

FIGURE 9. RRT Connect Algorithm with 2037 iterations and 2.28 seconds

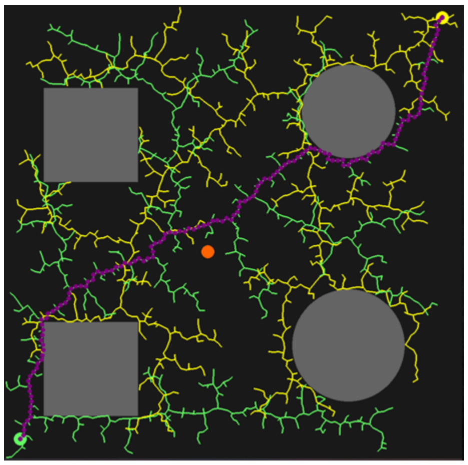

FIGURE 10. RRT Connect Algorithm with 1326 iterations and 1.45 seconds

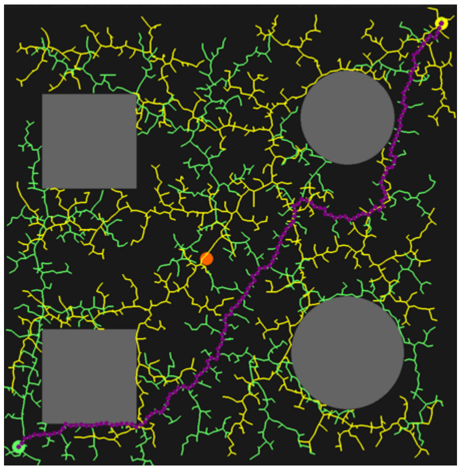

FIGURE 11. RRT Connect Algorithm with 1833 iterations and 2.13 seconds

The conventional RRT algorithm was very inconsistent in time consumption and the nodes expanded in random spots which are redundant.

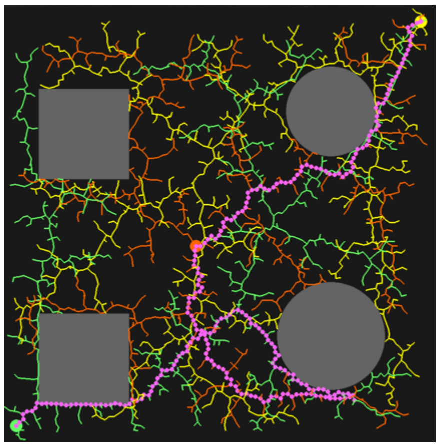

FIGURE 12. RRT Connect Algorithm (Third node improvement) with 1264 iterations and 1.7 seconds

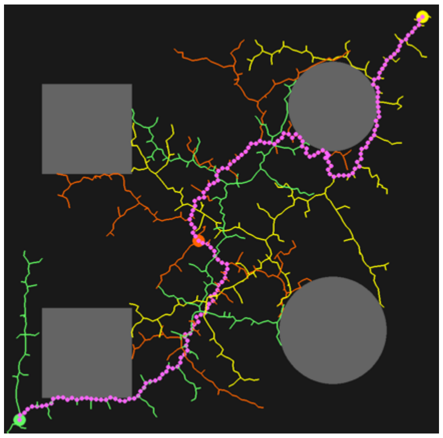

FIGURE 13. RRT Connect Algorithm (Third node improvement) with 533 iterations and 0.84 seconds

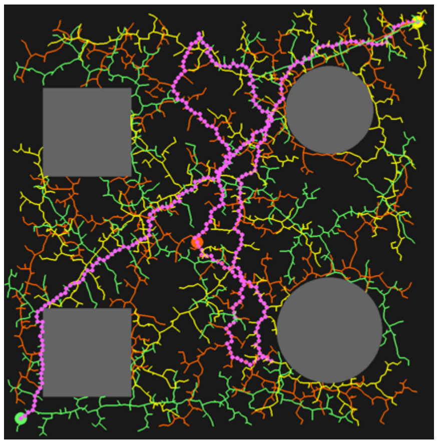

FIGURE 14. RRT Connect Algorithm (Third node improvement) with 1580 iterations and 2.89 seconds

The third node Improved RRT (IRRT) is faster compared to the conventional RRT but the path generated is long and complex. If cost is taken into account this improvement can be ineffective.

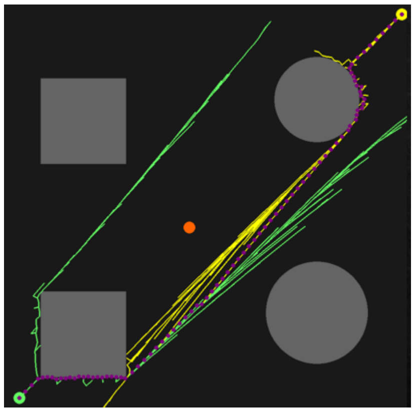

FIGURE 15. RRT Connect Algorithm (Guidance improvement) with 771 iterations and 0.91 seconds

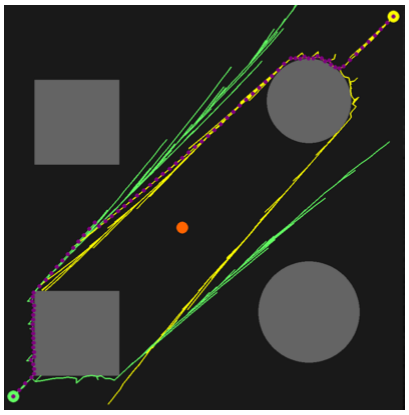

FIGURE 16. RRT Connect Algorithm (Guidance improvement) with 776 iterations and 0.91 seconds

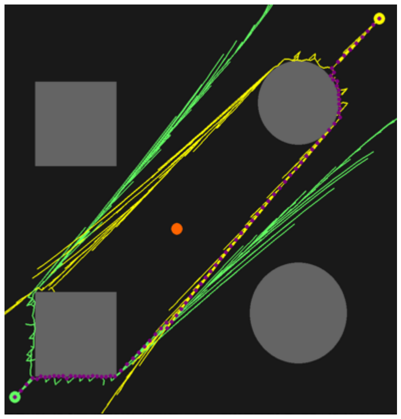

FIGURE 17. RRT Connect Algorithm (Guidance improvement) with 1418 iterations and 1.5 seconds

The Guidance Improved RRT (IRRT) is faster compared to the other RRT variants in this paper. This algorithm forms a straight line and overcomes the cost issue with the third node improvement. Both the start and end point start out in a straight line and hits the obstacles, and then gets stuck there and the conventional action set is called to steer the path away from the obstacles whose algorithm is mentioned in Figure 8.

Table: Average results of 50 experiments with the three algorithms

| Algorithms | Avg Time(secs) | Avg Iterations(times) |
|---|---|---|
| RRT Connect | 1.95 | 1170 |
| IRRT (3rd node) | 1.55 | 1120 |
| IRRT Guidance | 1.2 | 950 |

The conventional RRT connect algorithm on average took 1.95 seconds to execute in this environment, with 1730 iterations. The RRT Connect with the third node has an execution time of 1.55 seconds with 1120 iterations in the same environment. Finally, the RRT Connect with the guiding force has an execution time of 1.2 seconds with 950 iterations in this environment. By adding a simple third node, the execution time has been reduced by 20.5% and the number of iterations by 35.2% when compared to the original RRT Connect algorithm. When comparing the experimental data, it can be observed that the improved algorithm has a significantly lower number of iterations and a better path in the environment. Ultimately, the improved RRT Connect algorithm with the guiding force reduces the path-finding time by 38.4% and the number of iterations by 45% which properly correlates with the results from the research paper i.e. (36% and 51% respectively) when compared with the original RRT Connect algorithm. The RRT-Connect algorithm is not effective in smoothly navigating obstacles and the generated paths become more complex after the tree expansion encounters an obstacle. While both the RRT-Connect algorithm and the improved algorithm in this paper utilized a greedy expansion strategy, the new expansion strategy that was introduced involves adding nodes with a higher bias towards the target point and directional, resulting in straighter paths and the ability to quickly find a valid path after encountering obstacles. The improved RRT Connect with the guiding force algorithm obtains a path with a reduced length, path-finding time, and iteration count compared to the other two algorithms, in both a limited set of trials (10) and a larger set of trials (50). It's probable that this outcome is attributed to the fact that we have modified the spanning tree by employing a new node expansion technique that generates a node in the direction of the target at each iteration, which enhances the search rate of the algorithm. This modification, in combination with other algorithmic optimizations, could account for the superior performance of the improved algorithm.

## 5 Conclusion

This research paper focused on the RRT-Connect algorithm, which is a sampling-based path-planning algorithm that is effective in complex environments due to its quick search capability and simplicity. To enhance the algorithm's exploration efficiency, a third node is generated to expand the algorithm in a quadtree form. Additionally, to address the issue of RRT-Connect's blind search problem, the generation of new nodes is adjusted by increasing the bootstrap. As a result, the improved algorithm expands new nodes that prioritize the target point, thus accelerating the search efficiency. We conducted more than 50 experiments to verify the effectiveness of the improved algorithm. Overall, the improved RRT-Connect with the guiding force and the RRT-Connect with the third node algorithm reduce the number of iterations by 45% and 35.2%, the path-finding time by 38.4% and 20.5% compared to the traditional RRT-Connect algorithm. After implementing the improved RRT-Connect algorithm in Python, the improved algorithm performed very well in various aspects within a complex environment. The reason behind this success can be attributed to the third node generation and the increased bias towards the target point of the improved algorithm, which enhanced the exploration efficiency. These results correlate with the results and path generated from the research paper.

## 6 References

1. An Improved Path Planning Algorithm Based on RRT by QiongWei Zhang, LiaoMo Zheng, LunXing Li, BeiBei Li
2. RRT-based Path Planning for Car-Like Vehicles with Nonholonomic Constraints by Sotirios Spanogianopoulos
3. Path Planning Algorithm Based on the Improved RRT-Connect for Home Service Robot Arms by Shuyu Li, Donghui Zhao, Yizhen Sun, Junyou Yang, Shuoyu Wang
4. Review of wheeled mobile robots' navigation problems and application prospects in agriculture by X. Gao, J. Li, L. Fan, Q. Zhou, K. Yin, J. Wang, C. Song, L. Huang, and Z. Wang
5. MOD-RRT*: A sampling-based algorithm for robot path planning in a dynamic environment by J. Qi, H. Yang, and H. Sun
6. Rapidly-Exploring Random Trees: A New Tool for Path Planning by S. LaValle
7. Path planning of industrial robot based on improved RRT algorithm in complex environments by H. Zhang, Y. Wang, J. Zheng, and J. Yu