# Research on path planning based on improved RRT-Connect algorithm- A Review

Anbarasan Kandasamy
University of Maryland
College Park, MD

Manoj Kumar Selvaraj
University of Maryland
College Park, MD

## I. Introduction

Path planning algorithms in robotics serve an important role in helping autonomous navigation by generating collision-free paths from a robot's starting point to its goal. These algorithms are essential for a wide range of applications, including self-driving cars, industrial automation, and search and rescue missions. Path planning algorithms provide efficient and safe robotic movements by assessing environmental data and taking into account elements such as obstacles, landscapes, and dynamic changes. Potential fields, probabilistic roadmaps, and rapidly exploring random trees (RRT) are common techniques, each with unique strengths and adaptability for distinct contexts. Continuous developments in path-planning algorithms makes robotics forward toward safer and more efficient autonomous systems.

For applications such as autonomous driving and detecting systems, path planning is the process of determining a routine that avoids collisions between a starting point and a target point within a predetermined time limit. This is accomplished with the popular Extended Random Tree (RRT) technique, which does away with the requirement for intricate spatial modeling by sampling points in the state space and identifying collisions with barriers. RRT has limits even if it works well in resolving complex and high-dimensional constraint situations. Its random node expansion frequently results in long computation times and inferior solutions because it needs to try several times before finding a workable path. Furthermore, trees near places with obstacles develop more slowly due to fixed expansion steps, requiring lengthy detours.

Researchers from all around the world have suggested improvements to RRT to address these problems. Kuffner et al. 's RR T-Connect method accelerates algorithm convergence and increases planning efficiency by concurrently generating random trees from the start and target locations. The RRT-Connect algorithm is enhanced in this study to increase path quality and sampling efficiency. Target-biased sampling is first subjected to limitations to direct each sample in the direction of the target, eliminating pointless exploration and improving sampling efficiency. Furthermore, a shortcutting algorithm modifies the extended step length throughout the expansion phase of the process, smoothing the resulting path and accelerating convergence. As a result, the suggested approach shortens computing times, accelerates convergence, and raises the overall standard of the path's design.

## II. Literature Review

This study looks into ways to improve the RRT-Connect algorithm, which is an important part of path planning in robotics. RRT-Connect is widely applied for its ability to find collision-free pathways, but its efficiency and optimality have been questioned. To address these problems, the study suggests changes to the algorithm's methodology. The study specifically adds a target bias method to explore sampling points toward the objective, to speed up the exploration process.

Path planning is a emerging field in mobile robotics research, with applications ranging from robotic navigation and autonomous driving to unmanned aerial vehicles (UAVs) and more. Finding the collision-free path from the starting point to the destination is the main goal of path planning, with a focus on features like short travel distance and low travel time. Path planning uses number of algorithms, such as A*, D* and RRT. While, the A* and D* algorithms are excellent for local or real-time planning, they require a large amount of computer power, especially in three-dimensional spaces. RRT have strong search capabilities since it uses random sampling, yet its inherent randomness results in low search efficiency. Several improved algorithms, including RRT*, have been presented as solutions to the shortcomings of RRT.

The main goal of this paper is to enhance the RRT-Connect algorithm, which is a vital part of robotic path planning. The algorithm's shortcomings, with regard to its random sampling strategy, is the focus of the suggested improvements. The paper proposes to speed exploration by steering sampling points towards the target by using a target bias method. Furthermore, dynamic step length modifications during tree expansion are suggested to improve exploration velocity. The random tree is pruned using a shortcutting technique to smooth the resulting path. A comparative study is carried out to verify the effectiveness of the suggested improvements against the traditional RRT and RRT-Connect algorithms. Initial results show that the updated technique significantly reduces path length and execution time. These findings highlight how the improvements could further robotic path planning and allow for more dependable and effective autonomous navigation..

An important component of the proposed strategy is the dynamic modification of step lengths during tree expansion. By varying the step length, the algorithm attempts to achieve a balance between extensive exploration and efficient path discovery. This adaptive technique is intended to improve the algorithm's capacity to navigate complicated situations while reducing excessive processing costs.

## III. RRT- Rapidly Exploring Random Trees

The Rapidly Exploring Random Trees (RRT) path planning method has been introduced by Professor Lavalle. It received a lot of interest and use due to its ease of use, quick search speed, and efficiency in exploring complex and multi-dimensional environments. By randomly selecting points in Configuration Space, RRT builds a search tree sequentially. The starting point acts as the root node. The procedure entails creating random sampling points in space, locating each sample's closest search tree node, and determining whether there is an obstacle in the path. If there are no obstacles, new nodes are formed, extending the tree with a minimal step length in the direction of the sample point. Until the endpoint is attained or the maximum number of iterations is reached, this iterative process continues iterating. RRT is a randomized algorithm, therefore even with its efficiency, it might not always produce globally optimal pathways. But in complex environments, this property offers probabilistic completeness, guaranteeing that a workable solution is found after a sufficient number of planning iterations.

RRT builds a tree structure in configuration space by randomly sampling states and connecting them, in contrast to systematic exploration algorithms like A* or D*. RRT is appropriate for dynamic and complicated situations because of its randomized approach, which makes it possible to explore huge state spaces effectively. However, because RRT tends to ignore some regions or takes longer to converge to a solution, its reliance on random sampling can result in less-than-ideal pathways. Also, the connection of the environment may not be accurately reflected in the tree structure produced by RRT.

Various RRT enhancements and extensions have been presented in order to get around these limitations. These include RRT-Connect, which concentrates on direct connections between start and goal states, and RRT*, which optimizes the tree structure to find more efficient paths. These improvements provide improved functionality and performance for robotic path planning tasks by improving upon the fundamental ideas of RRT while resolving its shortcomings.

### A. Pseudocode

***Initialize*** *tree with start node*
***Declare*** *the obstacle space*
***Iterate*** *through required iterations:*
  *Sample random point in configuration space*
  *Find nearest node in tree to random point*
  *Extend tree towards random point*
  ***If*** *point is in obstacle-free path*
    *Add new node to tree*
    ***If*** *goal reached*
      ***END****: Backtrack path from start to goal*
  ***Iteration complete****: no path found within max iterations*

### B. Flowchart

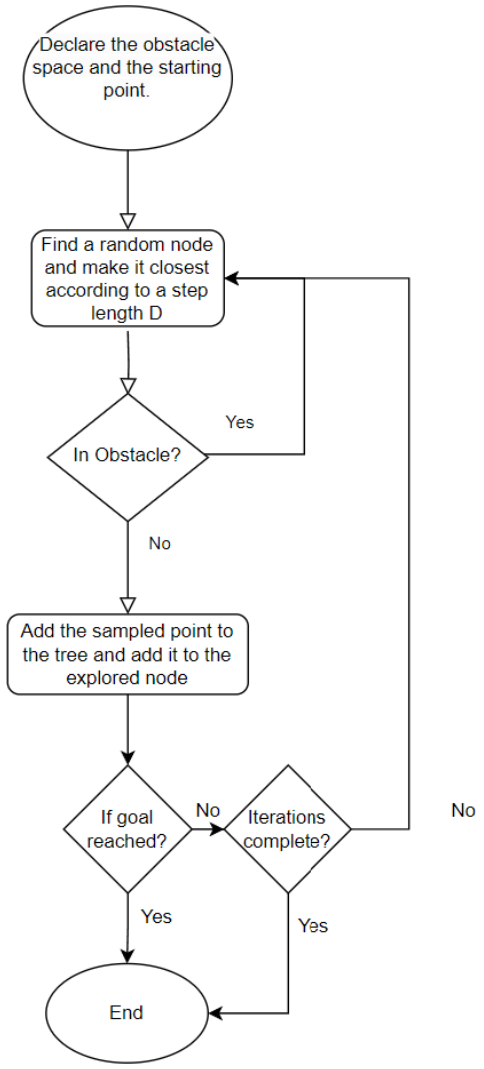



Fig. 1. RRT Flowchart

## IV. RRT Connect

Rapidly Exploring Random Trees (RRT) Connect algorithm is an extension of the original RRT algorithm. RRT Connect was first introduced to help with the difficulties of locating effective and collision-free routes in intricate settings. Since then, it has been extensively used in various industrial applications. This algorithm, which builds on the key concepts of RRT, provides an organized approach for computing paths by gradually creating two trees, one rooted at the start location and the other at the objective.

RRT connect is basically RRT but with root nodes from both the start and athe end point. Since RRT connects a bidirectional search, the convergence of the algorithm is quick and requires less iterations to find a solution.

The RRT-Connect algorithm defines two random trees within the non-obstacle space, each rooted at the starting point Q-init and the target point Q-goal respectively. These trees

undergo simultaneous expansion until they intersect, which is done by declaring an area threshold, signifying the discovery of a search path. As the random tree originating from Q-init explores the free space, another tree rooted at Q-goal is concurrently created and explored. Alternately, both trees generate new nodes, and each new node is computed against the other tree to determine if their Euclidean distance is less than a predefined step length p. When the distance between a pair of nodes is within p and no obstacles are blocking the connection between, the nodes are joined effectively merging the two random trees into a single tree and generating a path by backtracking. This iterative process of expanding and merging continues until the trees intersect, indicating the discovery of a feasible path between the start and goal points.

### A. Pseudocode

***Initialize*** *two points T_start, T_goal*
***Declare*** *the obstacle space*
***Iterate*** *through required iterations:*
  *Sample random point in configuration space*
  *Find nearest node in tree to random point*
  *Extend tree towards random point*
  ***If*** *point is in obstacle-free path*
    *Add new node to tree*
    ***If*** *trees intersect*
      ***END****: Backtrack path from start to goal*
    ***Swap*** *T_start, T_goal*
  ***Iteration complete****: no path found within max iterations*

### B. Flowchart

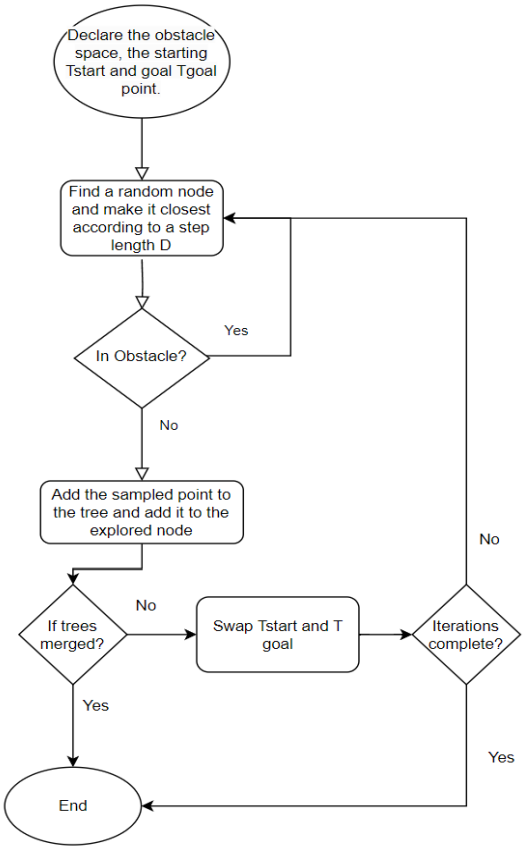



Fig. 2. RRT Connect Flowchart

## V. Improved RRT Connect

The improved RRT-Connect algorithm focuses on increasing the efficiency and quality of path planing. This is achieved by target biased sampling with constraints, which drives the sampling process towards the goal point while introducing restrictions to ensure that each sample approaches the target. This change avoids needless exploration of non-relevant areas, increasing the search towards the goal. Furthermore, the strategy involves managing random tree expansion, which allows for more targeted and efficient progress towards the goal. After creating the original path, the algorithm smoothes it with a shortcutting algorithm, reducing abrupt direction changes and enhancing path quality. Together, these changes result in faster convergence to the target, shorter runtimes, and higher-quality paths.

### A. Pseudocode

***Function sample****():*
  ***If*** *a random number is less than the goal bias:*
    *Return the goal position*
  ***Else****:*
    *Return a random point within the boundaries*
***Function nearest_node****(tree, sample_point):*
  *Initialize minimum distance as infinity*
  *Iterate through all nodes in the tree:*
    *Calculate the distance between the node and sample_point*
    ***If*** *distance is less than minimum distance:*
      *Set minimum distance as the current distance*
      *Set the current node as the nearest node*
  *Return the nearest node*
***Function new_node****(nearest, sample_point):*
  *Calculate the direction vector from nearest node to sample_point*
  *Normalize the direction vector and multiply by step size*
  *Calculate new node position as nearest node position + direction vector*
  *Create a new node at the new position*
  *Set nearest node as the parent of the new node*
  *Return the new node*
***Function extend****(tree, sample_point, explored_nodes):*
  *Find the nearest node in the tree to sample_point*
  *Create a new node in the direction of the sample point from the nearest node*
  ***If*** *the new node is within the bounds and doesn't collide with obstacles:*
    *Add the new node to the tree*
    *Add the new node to the explored nodes list*
    *Return the new node*
  *Return None*
***Function connect****():*
  ***For*** *max_iterations:*
    *Sample a random point*
    *Extend the start tree towards the sample point*
    *Extend the goal tree towards the sample point*
    ***If*** *new nodes are added to both trees:*
      *Calculate distance between the new nodes*

***If*** *the distance is small enough:*
*Link the start and goal trees*
*Return True*
*Return False*
***Function smooth_path****():*
*Use shortcutting to smooth the path by checking direct line segments between points*
*Set the smoothed path as the final path*

The explanation of pseudo-code is as follows,
**1.** The sample function generates a random point either within the goal bias or within the boundaries of the area.

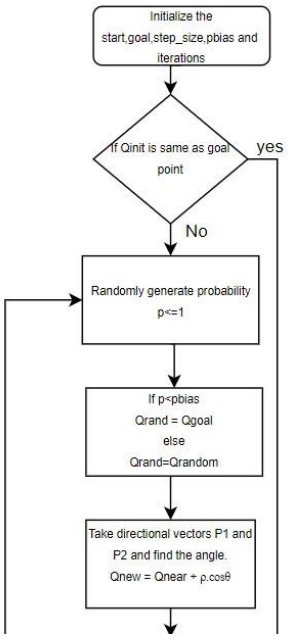
p←random(0,1)
if (p<pbias)
Qrand = Qgoal
else
Qrand = Qsamp

**2.** The nearest_node function finds the closest node in the tree to a sample point by calculating the distance to each node.
**3.** The new_node function calculates a new node position towards the sample point from the nearest node and creates a new node.

$P1 = Qnear - Qrand$
$P2 = Qnear - Qgoal$
$\cos\theta = (P1.P2) / ||P1||||P2||$
$Qnew = Qnear + \rho.\cos\theta$

P1 and P2 are directional vectors. P → step_size
**4.** The extend function extends a tree towards a sample point, creating a new node if there are no collisions and within bounds.
**5.** The connect function aims to link the start and goal trees by sampling points and extending each tree towards the sample point.
**6.** The smooth_path function optimizes the final path using shortcutting, which checks direct line segments between points.

*B. Flowchart*



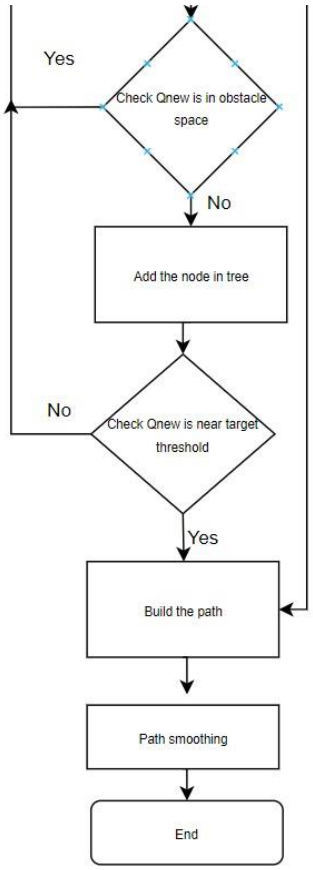




Fig. 3. Improved RRT Connect Flowchart

*C. Adaptive Step Adjustment*

The RRT-Connect algorithm is a pathfinding algorithm that uses a random tree to explore a barrier-free (open) space to find a path from a start point to a goal point. The algorithm typically uses a fixed step length, denoted as ρ, to control how far the tree extends when exploring the space.

The approach described proposes an adaptive step adjustment, where the step length changes based on whether the tree is able to make progress (i.e., advance) without hitting an obstacle, or if it encounters an obstacle.

Here's how the adaptive step adjustment works:
1. **Initial Step Length:** Begin with an initial step length of ρ.
**2. Expanding the Tree:** When the tree expands by adding a new node.
If the expansion is successful (no collision with obstacles and not connecting to another tree) and when the direction vectors are oriented in same direction.

**Increase the step length:** If the expansion successfully adds a new node, the step length is increased by one ρ for the next expansion. This allows the tree to explore larger areas more quickly.
If the expansion fails (a collision occurs during expansion)

**Reset the step length:** If the expansion hits an obstacle or the direction vectors are lying in opposite direction,

the step length is reset to the original ρ. This ensures more precise and controlled exploration in the presence of obstacles.
**3. Continue Expanding:** The tree continues to expand in this manner, adjusting the step length according to whether it is able to successfully advance or encounters an obstacle.
By adapting the step length in this way, the algorithm can explore the space more efficiently.

### *D. Path Smoothing*

The shortcutting smoothing technique reduces unnecessary points and creates shortcuts between them when possible. It starts with the first point in the path and iterates through the remaining points, checking if there is a direct line that can be drawn from the current point to a later point in the path without any obstacles in the way. This check is done using a method that determines if the direct connection is possible.

If the direct connection is valid, the method skips over the points in between and moves directly to the later point. This process is repeated as the method progresses through the path, adding only the significant points (such as the starting point, ending point, and points where a direct connection cannot be made) to a new, smoother path. Once the entire original path has been processed, the simplified path replaces the original path. This approach helps in making the path more direct and efficient, reducing the complexity of navigating it.

### *E. Advantages:*

**Exploration:** This refers to the process of searching through the space without specific bias towards any particular region. In the RRT-Connect algorithm, this is akin to randomly sampling points in the free space around the tree nodes. Exploration allows the algorithm to discover new potential paths and avoid getting stuck in local minima.

**Exploitation:** This refers to focusing on a specific target or region of interest in the search space, leveraging prior knowledge or an estimated direction towards the goal. In the goal-biased approach, the algorithm introduces a reference value (pbias) to guide the tree expansion towards the goal point more frequently. By generating random points closer to the goal, the algorithm can more effectively hone in on a path towards the goal, thus exploiting the knowledge of the desired target.

The goal-biased approach in the RRT-Connect algorithm integrates exploration and exploitation to enhance the efficiency of the search process. Goal-biased sampling is achieved by setting a probability threshold (pbias). When a random number (p) is generated during sampling, if it is below pbias, the algorithm targets the goal point, effectively focusing the search in the direction of the goal. This form of exploitation helps guide the search process closer to the desired end point, thus improving the algorithm's convergence rate.

At the same time, introducing a direction vector from the random point towards the goal point influences the expansion direction of the new nodes. This additional directional constraint ensures that the tree growth is not only oriented towards the target, but also tends to navigate around obstacles in a more efficient and direct manner. By combining random sampling for exploration with goal-oriented biasing for exploitation, the algorithm strikes a balance between discovering new potential paths and directing the search efficiently towards the goal. This approach maximizes the chance of finding a feasible and smooth path while maintaining robustness in varying environments.

## VI. Results

The study published results comparing three path planning algorithms: RRT, RRT-Connect, and Improved RRT-Connect. These algorithms were tested on a complex map with obstacles in a 500x500 grid. The findings likely highlight differences in performance regarding speed, efficiency, and success rate of pathfinding.

In the fig. 4., black indicates obstacles that block the path of a path planning algorithm. Orange marks the start point, where the path should begin its journey. Green is the goal point, representing the destination the path must reach. The algorithm needs to navigate from the orange start point to the green goal point while avoiding the black obstacles. This setup tests the algorithm's ability to find a viable path through a complex environment.

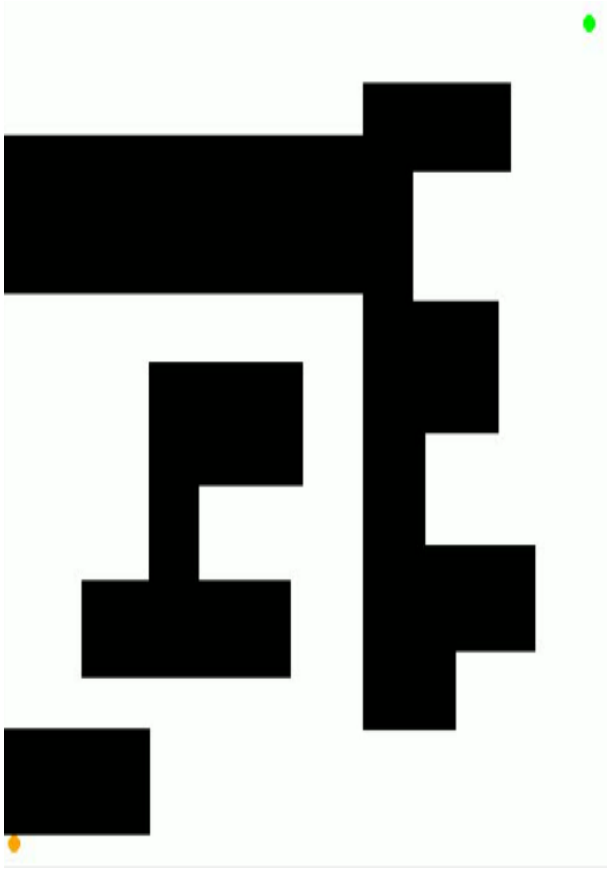


Fig. 4. Complex Environment with Obstacles

The following figures represents Node exploration and final path after smoothing for the algorithms RRT, RRT-Connect and Improved RRT-Connect respectively.

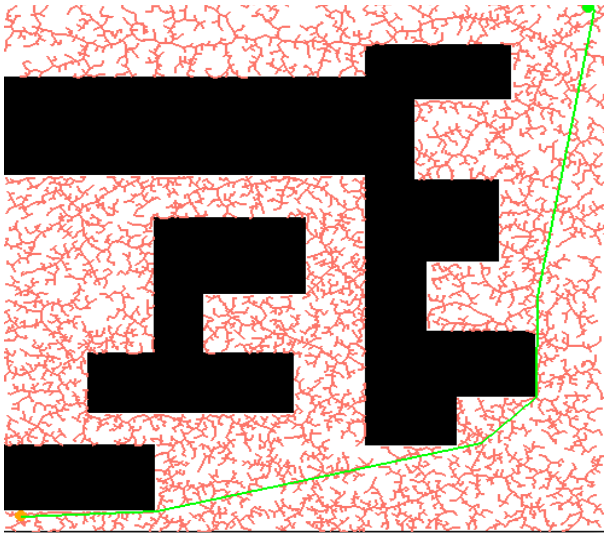


Fig. 5. RRT Final Path

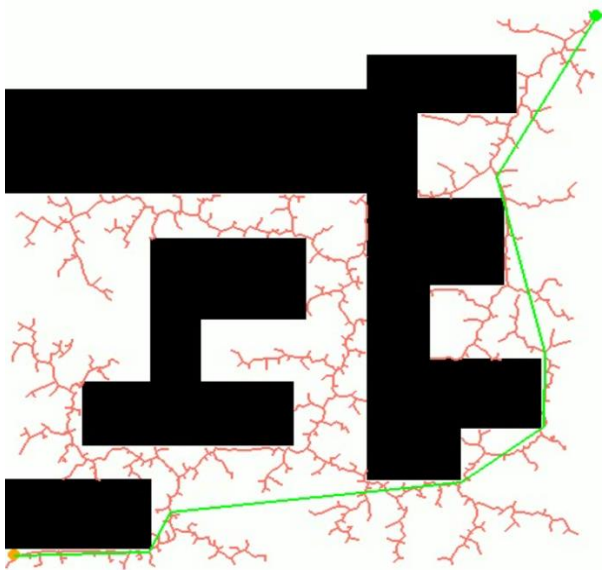


Fig. 6. RRT-Connect Final Path

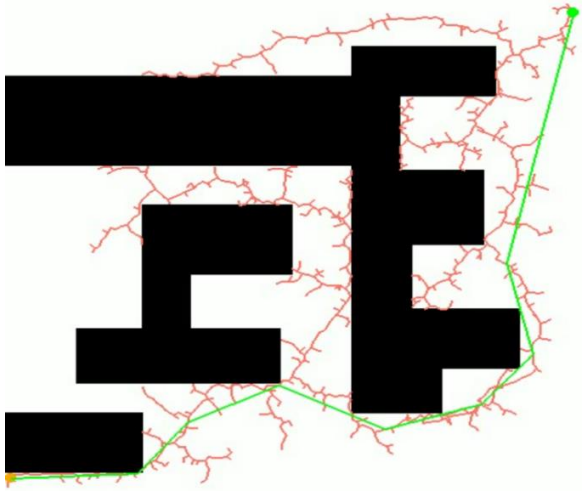


Fig. 7. Improved RRT-Connect Final Path

In path planning, RRT explores more nodes compared to RRT-Connect due to its standard tree expansion without specific optimization strategies. RRT-Connect introduces a bidirectional search that connects two trees, which reduces the number of nodes explored by focusing on efficiently finding a path. Improved RRT-Connect further optimizes the search process through exploiting the nodes towards the goal (goal-biased sampling) and adaptive step size helps the node to reach goal point quicker. These improvements focus on refining the search and tree connection, resulting in fewer nodes explored compared to RRT-Connect. Overall, the progression from RRT to RRT-Connect and then to Improved RRT-Connect reflects increasing efficiency in the search process and path planning.

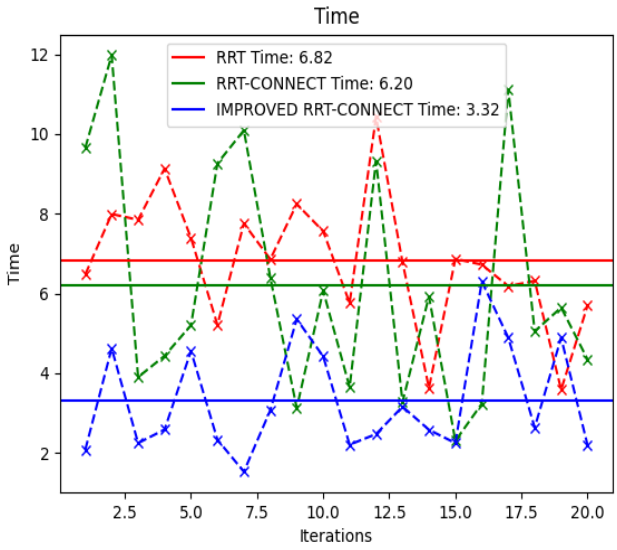



Fig. 8. Time Comparision

The Rapidly-exploring Random Tree (RRT) algorithm is a path planning method that constructs a tree from the initial point by randomly sampling points in the search space and expanding the tree towards them. This can result in longer times to find a path as the approach doesn't prioritize a specific direction or focus on the goal. RRT Connect improves upon RRT by using two trees: one starting from the initial point and the other from the goal point, aiming to connect them as soon as possible. This bias toward connecting the trees generally speeds up the process compared to the original RRT. Improved RRT Connect takes this efficiency even further by being goal-biased, actively steering towards the goal point. This focus on the target allows it to reduce unnecessary exploration and improve the efficiency of the search. In your data, this results in a performance improvement of 46.45% compared to RRT Connect, making Improved RRT Connect the most effective algorithm in terms of speed and efficiency.

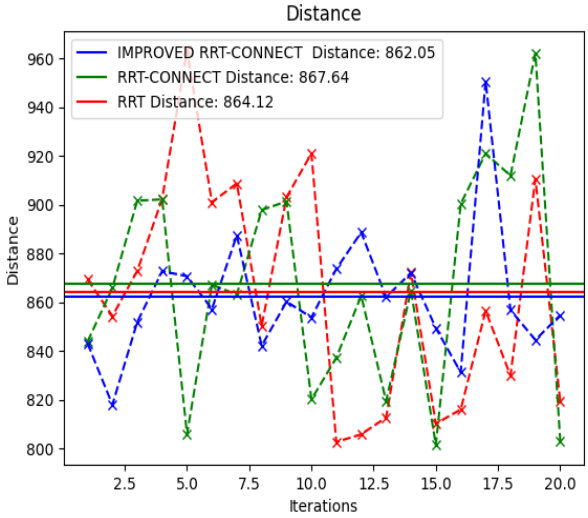



Fig. 9. Optimal Path Comparision

When comparing the algorithms RRT, RRT Connect, and Improved RRT Connect over 20 iterations, particularly in terms of the distance after path smoothing, we can observe distinct differences in their behavior and performance. RRT is often less efficient due to its randomized exploration, leading to longer, more circuitous paths and potentially requiring more iterations. RRT Connect builds upon RRT by connecting trees from both the start and goal points, resulting in faster convergence and generally more direct paths. However, the algorithm might yield slightly higher distances compared to RRT after smoothing because it prioritizes connecting to the closest point in the opposite tree, which may not always correspond to the most direct route.

Improved RRT Connect stands out as the most efficient algorithm in this comparison, converging quickly to find shorter and smoother paths with fewer iterations. Its optimizations streamline the path-finding process and enhance the connection between trees, leading to shorter distances and more direct paths. Though RRT Connect shows improvement over RRT, there is a close similarity in their performance, with RRT Connect potentially producing slightly longer paths due to its approach to connecting trees and complex environment. Improved RRT Connect performs better in terms of efficiency and directness in path-finding.

## VII. Conclusion

In conclusion, the analysis of the path planning algorithm RRT, RRT Connect, and Improved RRT Connect highlights the progression from the basic RRT approach to more advanced methods that increase efficiency and speed in finding a viable path through complex environments. While the original RRT algorithm provides a foundation for path planning, it suffers from longer times to find a path and inefficient, meandering routes due to its randomized exploration strategy. RRT Connect improves upon this by using a bidirectional search that connects trees from the start and goal points, leading to faster convergence and more direct paths. However, it may produce equal or slightly longer distances after path smoothing due to its priority of connecting the closest points in the opposite trees and complex environment.

Improved RRT Connect, on the other hand, emerges as the most efficient algorithm of the three. By employing goal-biased sampling and adaptive step sizes, it streamlines the path-finding process and enhances the connection between trees, resulting in quicker convergence and shorter, smoother paths. This method outperforms RRT Connect by 46.45% and sets a new standard for speed and efficiency in path planning. Overall, the findings demonstrate the value of advanced strategies in optimizing path planning algorithms, with Improved RRT Connect showing the most promise for real-world applications where fast and efficient navigation is essential.

## VIII. Reference

**[1]** H. Yang, H. Li, K. Liu, W. Yu and X. Li, "Research on path planning based on improved RRT-Connect algorithm," 2021 33rd Chinese Control and Decision Conference (CCDC), Kunming, China, 2021, pp. 5707-5712

**[2]** S. Li, D. Zhao, Y. Sun, J. Yang and S. Wang, "Path Planning Algorithm Based on the Improved RRT-Connect for Home Service Robot Arms," 2021 IEEE International Conference on Intelligence and Safety for Robotics (ISR), Tokoname, Japan, 2021, pp. 403-407

**[3]** J. Chen, Y. Zhao and X. Xu, "Improved RRT-Connect Based Path Planning Algorithm for Mobile Robots," in IEEE Access, vol. 9, pp. 145988-145999, 2021

**[4]** D. Zhang, Y. Xu and X. Yao, "An Improved Path Planning Algorithm for Unmanned Aerial Vehicle Based on RRT-Connect," 2018 37th Chinese Control Conference (CCC), Wuhan, China, 2018, pp. 4854-4858

**[5]** Y. Zhu, Y. Tang, Y. Zhang and Y. Huang, "Path Planning of Manipulator Based on Improved RRT-Connect Algorithm," 2021 2nd International Conference on Big Data & Artificial Intelligence & Software Engineering (ICBASE), Zhuhai, China, 2021, pp. 44-47.