Optimized Double Tree RRT* Algorithm for Mobile Robots

Datta Lohith Gannavarapu
M.Eng. Robotics
University of Maryland
College Park, USA
gdatta@umd.edu

Venkata Sai Sricharan
M.Eng. Robotics
University of Maryland
College Park, USA
charan03@umd.edu

Rishabh Namdev
M.Eng. Robotics
University of Maryland
College Park, USA
rishabh@umd.edu

***Abstract*— Motion planning is crucial for enabling robotics and autonomous systems to navigate complex environments. The Rapidly Exploring Random Trees (RRT) algorithm has proven to be effective in finding feasible paths but suffers from slow convergence in large configuration spaces or when start and goal configurations are distant. In this paper, we propose an Optimized version of Double Tree RRT* algorithm, an extension of RRT* that incorporates a bidirectional double-tree approach to enhance efficiency. The Double Tree RRT* algorithm maintains two separate trees, a forward tree growing from the start configuration and a backward tree growing from the goal configuration. These trees simultaneously explore the configuration space using random sampling and extension operations.**

Keywords— Motion planning, RRT algorithm, RRT* algorithm, bidirectional search, double-tree approach, star-based optimization, robotics, autonomous systems.

I. INTRODUCTION

A. Path Planning

Path planning, also known as motion planning or trajectory planning, is a fundamental problem in robotics and autonomous systems. It involves the task of determining a feasible and optimal path for a robot or agent to navigate from a starting point to a desired goal point in each environment, while avoiding obstacles and adhering to constraints. Path planning algorithms typically operate in a configuration space, which represents the set of all possible states the robot can be in. The configuration space can be high-dimensional, capturing the robot's position, orientation, and other relevant variables. The environment is usually represented as a map or occupancy grid, where obstacles and free space are defined.

Path planning techniques often face challenges of increased computational time when dealing with larger maps or complex environments. Traditional methods, such as graph search algorithms or potential fields, may struggle to scale efficiently due to the exponential growth of the search space. Rapidly Exploring Random Trees (RRT) algorithms were introduced as a solution to reduce computational time by using random sampling and incremental tree expansion. RRT quickly explores the configuration space, but it does not guarantee optimal paths. However, it provides a good trade-off between computational efficiency and path quality.

To improve the optimality of paths, RRT* was developed by incorporating the concept of optimization. RRT* optimizes paths by iteratively rewiring the tree structure and considering a cost function to find the lowest-cost path. Although RRT* provides more optimal paths, it can still be computationally expensive for larger maps due to the extensive rewiring process.

To further enhance efficiency without compromising optimality, the Double Tree RRT* algorithm was introduced. Double Tree RRT* takes a bidirectional approach, expanding trees simultaneously from both the initial and goal nodes. This bidirectional random sampling strategy enables the algorithm to converge on optimal paths more efficiently by exploring the search space from two directions. As a result, Double Tree RRT* produces optimal paths with significantly reduced computational time compared to other planning techniques.

B. Optimality

Optimality in path planning refers to the quality of a generated path based on specific criteria or objectives while considering the computational time factor. It represents how well the path satisfies desired goals and meets certain criteria within feasible computational limits. The goal of path planning is to find a trajectory or sequence of states that satisfies certain criteria, such as minimizing travel time, energy consumption, or avoiding collisions. The path should be smooth, continuous, and feasible for the robot's dynamics and capabilities. The bidirectional nature of the algorithm allows it to exploit the configuration space's structure, resulting in faster convergence towards a solution. In addition to bidirectional search, the Double Tree RRT* algorithm incorporates the star-based optimization technique from RRT*. This optimization biases the tree expansion towards promising regions of the configuration space, improving convergence rates and reducing overall path length. To evaluate the algorithm's performance, we compare Double Tree RRT* with the state-of-the-art RRT* algorithm in various simulated scenarios featuring complex environments and high-dimensional spaces. The experimental results demonstrate that our proposed algorithm outperforms RRT* in terms of convergence time and path quality. The bidirectional double-tree approach significantly reduces exploration effort, while the star-based optimization further enhances the solution quality. The Double Tree RRT* algorithm represents a valuable contribution to the field of motion planning, providing an efficient and effective approach for path finding in robotics and autonomous systems. Its ability to handle large configuration spaces and distant start-goal configurations makes it particularly suitable for real-world applications. The experimental results highlight its potential for improving the performance and reliability of autonomous systems in challenging environments.

II. BACKGROUND

Path planning algorithms have long been faced with the challenge of finding an optimal or feasible path within a reasonable computational time. While various techniques have been developed over the years, the Rapidly Exploring Random Trees (RRT*) algorithm emerged as a solution to improve efficiency. RRT* addresses the computational time issue by employing a randomized sampling-based approach that quickly explores the configuration space. By constructing a tree-like structure, RRT* efficiently expands toward unexplored regions, enabling faster path planning. However, to further enhance efficiency, the Double Tree RRT* algorithm was introduced. This algorithm utilizes two trees, one forward and one backward, allowing for a more thorough exploration of the search space. By simultaneously expanding from both the start and goal configurations, Double Tree RRT* significantly improves efficiency, reducing the computational time required to find optimal or near-optimal paths. This advancement has made path planning more efficient, enabling applications in real-time scenarios and domains where computational time is a critical factor.

A. Literature Review

In [1] Long Chen, Yunxiao Shan, Wei Tian, Bijun Li, and others proposes the Double Tree Rapidly Exploring Random Graph (DRRG) algorithm, which uses two trees to search the configuration space, aiming for faster convergence and improved search efficiency compared to the traditional RRT* algorithm. The authors demonstrate the effectiveness of DRRG in simulations and real-world experiments, emphasizing its potential for mobile robotic vehicle applications.

In [2] the authors introduce a new variant of RRT called Intelligent Bidirectional-RRT* (IB-RRT*), which is an improved variant of the optimal RRT* and Bidirectional-RRT* (B-RRT*) algorithms. It is specifically designed for complex cluttered environments. IB-RRT* utilizes the bidirectional trees approach and introduces an intelligent sample insertion heuristic for fast convergence to the optimal path solution using uniform sampling heuristics. Experimental results demonstrate the superior efficiency of IB-RRT* in comparison with RRT* and B-RRT in complex cluttered environments.

In [3] Binghui Li, Badong Chen, and others present an improved planning method based on RRT-Connect and RRV called Adaptive RRT-Connect (ARRT-Connect). This method is designed to overcome the inherent difficulty of sampling-based methods in solving planning problems with narrow passages. The proposed ARRT-Connect algorithm can efficiently plan paths in various environments, including those with narrow passages, while still maintaining the ability of RRT algorithms to plan paths in other environments. The planner is adaptable to a variety of environments and can accomplish path planning in a short time.

In [4] Zongyuan Shen, James P. Wilson, Ryan Harvey, and Shalabh Gupta propose a novel algorithm called MRRT (Multiple Rapidly-Exploring Random Trees) for fast online replanning in dynamic environments with moving obstacles. This algorithm is built upon the RRT algorithm with a multi-tree structure and is designed to handle unknown static obstacles and moving obstacles.

In [5] Jonathan, D. Gammell,, Siddhartha, S. Srinivasa, and Timothy D. Barfoot, the authors propose an algorithm called Informed RRT* that improves the efficiency of optimal sampling-based path planning. Informed RRT* focuses the search for the optimal path by directly sampling from an admissible ellipsoidal heuristic.

In [6] Yifeng Huang, Kamal Gupta propose a novel approach that combines Rapidly-Exploring Random Trees (RRT) and Simultaneous Localization and Mapping (SLAM) techniques. The RRT-SLAM planner integrates uncertainty handling mechanisms within the RRT-based motion planning framework. The authors address the limitations of traditional RRT algorithms and emphasize the importance of incorporating SLAM techniques for robust motion planning. The proposed methodology involves integrating SLAM techniques with the RRT algorithm, resulting in improved exploration efficiency, map accuracy, and robustness to uncertainties. The paper highlights the significance of integrating uncertainty handling mechanisms within the RRT-based framework for effective robot exploration in uncertain environments.

In [7] by Hai Zhu Pan and Jin Xue Zhang authors introduce an extension to the Rapidly Exploring Random Trees (RRT) algorithm that incorporates Simultaneous Localization and Mapping (SLAM) techniques. The proposed extended RRT algorithm enables the robot to build a map of the environment while simultaneously planning its motion. By integrating SLAM capabilities, the algorithm improves the accuracy and reliability of the generated motion plans. The extended RRT algorithm actively updates the map and refines the robot's knowledge of the environment during the planning process, resulting in more informed and efficient motion plans. The paper provides evidence of the improved performance of the approach through simulations or real-world experiments, showcasing the benefits of integrating SLAM techniques into the RRT algorithm for enhanced robot motion planning in unknown or dynamic environments.

III. TERMINOLOGY

A. Definitions of Related Terminology

In the following, we define the *Occupancy Grid Map* as ***X***, the varying length of edges as *Control Input*, ***Q***, *rewire_count* is the number of nodes to consider for rewiring or optimal path, the initial pose of the mobile robot as *x_init* and the goal pose as *x_goal*, *trees* as a list of trees. ***X_dimensions*** *and* ***Obstacles*** are parameters which initialize the Occupancy Grip map. Besides these definitions we also use the following terminology.

steer() – Steers the tree towards the goal by Euclidean bias.

new_and_near() – Gives the nearest new node for a varying valid control input steered towards the goal for a tree.
connect_trees() – Connects the trees by adding a new edge and a vertex if the shortest path is found.
swap_trees() – Swaps the processing of trees and the initial and goal poses for both trees to grow.
rewire() – Rewires the original tree by updating the parent of a nearby vertex if a shorter feasible path is found.
reconstruct_path() – Reconstructs the path after the two trees are connected.

IV. Double-Tree RRT* Approach and SLAM

A. Double Tree RRT*

In the traditional RRT* algorithm, a single tree is grown from the initial configuration towards the goal configuration. The tree expands by randomly sampling points in the configuration space and connecting them to the nearest point in the tree, gradually exploring the space. However, this can sometimes result in inefficient paths, especially when the goal is located far away from the initial configuration. The bidirectional RRT* approach addresses this limitation by growing two trees simultaneously—one from the initial configuration and the other from the goal configuration. These trees grow towards each other until they meet, forming a path from the initial configuration to the goal configuration. This bidirectional exploration allows for a more efficient exploration of the configuration space, reducing the search effort.

The bidirectional (double tree) approach in RRT* offers several advantages over the traditional single-tree approach. Firstly, it improves search efficiency by simultaneously growing two trees from the start and goal configurations. This bidirectional exploration allows the algorithm to explore the configuration space from both directions, reducing the search effort and converging towards a solution faster. Secondly, the bidirectional approach enhances the quality of the generated paths. By exploring the space from both ends, the algorithm is more likely to capture the shortest and most feasible paths between the start and goal configurations. This leads to improved path quality and optimality compared to the single-tree RRT*. Moreover, bidirectional exploration increases the success rate of finding a feasible path, especially in complex or cluttered environments. By searching for paths from both sides, the algorithm increases the probability of finding a valid connection between the trees, thus increasing the chances of finding a solution.

Additionally, the bidirectional RRT* approach demonstrates flexibility in handling dynamic environments. In scenarios where obstacles or the goal position may change over time, the algorithm can adapt by rerouting and exploring new areas. By growing the trees in a bidirectional manner, the algorithm can effectively respond to environmental changes and find alternative paths if necessary. Random Exploration of the nodes from *x_init and x_goal.*

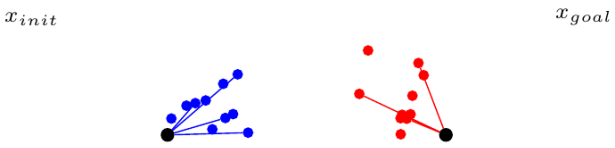

Figure 1: Example of Double Tree RRT* Exploration
The tree are connected to each other when they appear to meet in a bound radius search area.

B. SLAM (Simultaneous Localization and Mapping)

SLAM is a vital technique used in robotics and autonomous systems to simultaneously create a map of an unknown environment and determine the robot's position within it. When it comes to path planning, SLAM plays a crucial role by providing map information and localization estimates. The process starts with acquiring data from various sensors like cameras, LIDAR, or radar. The acquired data is then processed to extract significant features such as landmarks or distinctive points. Using these features and sensor measurements, SLAM algorithms create a map representation of the environment while estimating the robot's pose. With an accurate map and reliable localization, path planning algorithms come into play to find an optimal path considering obstacles, constraints, and the robot's capabilities. Once the path is planned, the robot executes the trajectory, adjusting its movements based on real-time perception and SLAM updates. The integration of SLAM and path planning enables robots to autonomously navigate unknown or changing environments, making informed path decisions and adapting to dynamic conditions.

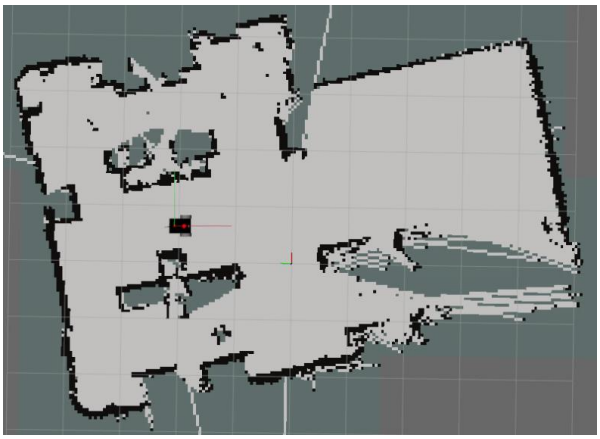

Figure 2: Example of SLAM Mapping

V. Proposed Double Tree RRT*

The proposed Double Tree RRT* algorithm uses varying edge lengths in every iteration confining to the number of maximum samples to be explored until the trees connect, this is achieved by the control input ***Q***, which is given to the algorithm and this control input is applied to both the trees which arise from start and goal node. The ***steer*** function which is used in this algorithm directs the tree towards the goal. For a tree exploring from the initial pose, ***x_init*** *is*

directed towards the goal and the tree exploring from the goal pose, ***x_goal*** is directed towards the initial pose for every explored random node. In this algorithm, varying edge lengths are defined as a step size to be considered for the nearest node which are generated after applying the steer function. The probability of connecting two trees from both ends is high when applied with these functions and this generates a near to optimal feasible path with less time complexity. The two trees are continuously checked for a connection and if a connection is possible, the shortest path between nodes is selected avoiding the obstacles in an occupancy grid map, and the searching for the path ends.

Algorithm 1: Optimized DT-RRT*

Input: Initial and goal configuration x_{init} and x_{goal}, number of samples *max_samples*, control inputs Q, rewiring count *rewire_count*
Output: Best path σ_{best}
Initialize V_0 with x_{init} and E_0 with None, V_1 with x_{goal} and E_1 with None, c_{best} with infinity, σ_{best} with None, swapped with False
while *True* **do**
  for *each control input q in Q* **do**
    for *i in range(q[1])* **do**
      x_new, *x_nearest* = new_and_near(0, *q*)
      if *x_new is not None* **then**
        L_near = get_nearby_vertices(0, x_{init}, *x_new*)
        connect_shortest_valid(0, *x_new*, *L_near*)
        if *x_new in* E_0 **then**
          rewire(0, *x_new*, *L_near*, *rewire_count*)
          L_near = get_nearby_vertices(1, x_{goal}, *x_new*)
          connect_trees(0, 1, *x_new*, *L_near*)
          c = get_cost(*x_new*)
          if $c < c_{best}$ **then**
            $c_{best} = c$
            σ_{best} = get_path(0, *x_new*) + get_path(1, *x_new*)
          end
        end
        if *samples_taken* $\geq$ *max_samples* **then**
          unswap()
          if σ_{best} *is not None* **then**
            return σ_{best}
          end
          else
            return *None*
          end
        end
      end
    end
    swap_trees()
end

The Occupancy Grip Map is a Search Space created for the path to be explored with the presence of obstacles. The probability of finding an Obstacle in a given position is 1 and the free space is 0. This Occupancy Grip Map is created further based on the Occupancy Map generated by the SLAM using LIDAR data to be integrated to implement the ***DT-RRT**** algorithm for a mobile robot to move freely avoiding the obstacles.

The ***obstacle_generator()*** function takes the input ***O(List of Obstacles)*** along with the dimensions of the map and generates the obstacles on the Occupancy Grid Map as mentioned in the ***Algorithm 2***.

The Proposed DT-RRT* algorithm works well in compact space maps, in which RRT* generally fails to find a path with less time complexity. The varying length edges, which are given as control inputs, generate new nodes which can find paths in compact and narrow maps as well. The Path generated is random but near to optimal in nature with very less time complexity.

Algorithm 2: Occupancy Grid Map

```
Input: dimension_lengths, O=None
Output: OccupancyGrid instance
if length of dimension_lengths < 2 then
  | Exception: "Must have at least 2 dimensions"
end
dimensions ← length of dimension_lengths;
dimension_lengths ← dimension_lengths;
p ← index.Property();
p.dimension ← dimensions;
if O is None then
  | obs ← index.Index(interleaved=True, properties=p);
end
else
  | obs ← index.Index(obstacle_generator(O), interleaved=True,
  |   properties=p);
end
```

VI. HARDWARE IMPLEMENTATAION & RESULTS

In this study, we implemented the Double Tree RRT* algorithm for path planning on a TurtleBot3 Burger robot. By integrating the algorithm with Simultaneous Localization and Mapping (SLAM) techniques, including G-mapping, we aimed to enhance mapping, localization, and path planning capabilities. The experiments were conducted in a simulated environment, and various scenarios featuring complex environments and high-dimensional spaces were defined to test the algorithm's performance. To evaluate the effectiveness of the Double Tree RRT* algorithm with SLAM integration, we compared the theoretical path produced by the algorithm with the actual path followed by the TurtleBot3 Burger robot. By extracting the coordinates from the odometry data of the robot, we accurately assessed the path taken. Plotting the theoretical path and the robot's path on a graph allowed us to analyze the algorithm's ability to navigate the environment accurately.

Figure 3: TurtleBot Burger

The TurtleBot's position is given as the start pose(*x_init*) and the end location as end pose(*x_goal*). The path is generated and a closed loop controller as depicted in Figure 4.

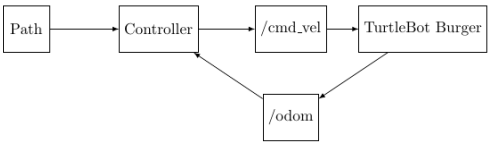

Figure 4: Closed Loop Controller

The Results for the path generated between two points for the TurtleBot to move freely are depicted in the figures below.

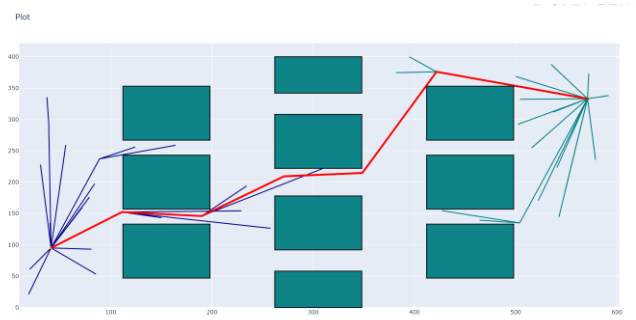

Figure 5: Optimized DT-RRT*(Control input - [40,40]) Time – 0.6s | Samples – 194

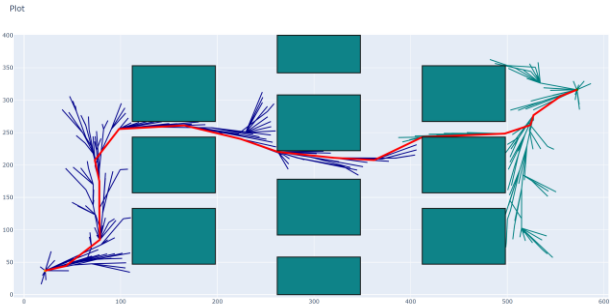

Figure 6: Optimized DT-RRT*(Control input - [8,8]) Time – 1.1s | Samples – 566

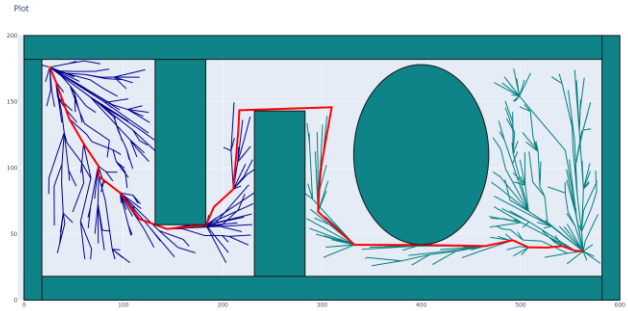

Figure 7:Optimized DT-RRT*(Control input - [8,8]) Time – 1.6s | Samples – 732

By observing the above results with a clearance of 200mm for the robot to move freely, the path generated by the two trees is near to optimal and feasible. The varying control inputs significantly reduces the computation time and number of samples taken to explore the path as observed in Figure 4 and Figure 5. In narrow and compact map such as Figure 6, the path is generated with time to explore being 1.6 seconds. In practical scenarios, the need to find a path consuming less time would be more advantageous and the proposed DT-RRT* algorithm solves this issue with better results than other path planning algorithms.

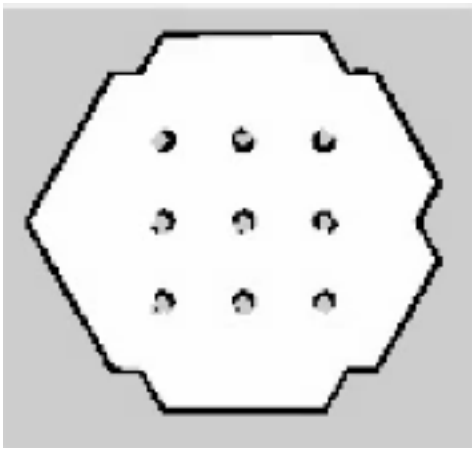

Figure 8: Occupancy Map Generated through SLAM

Using the LIDAR data from the TurtleBot burger, a Occupancy Map is generated with the help of SLAM (Figure 8). The proposed algorithm is integrated to generate path in the unknown environment mapped through SLAM, this is depicted in Figure 9.

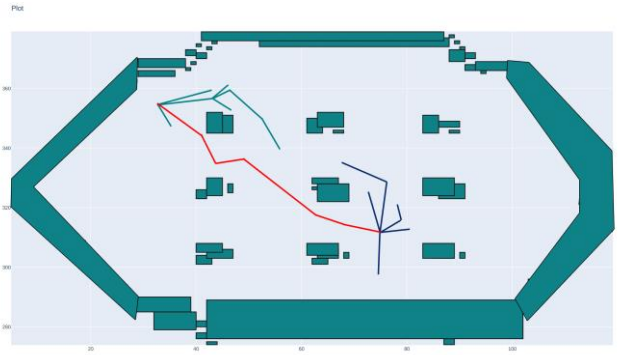

Figure 9: Path between two positions in an unknown environment (Map from SLAM)

The TurtleBot with the help of LIDAR is rotated in an unknown environment and using SLAM, this LIDAR data is used to map the Occupancy grid as visualized in ***Figure 8***. This map represents the obstacles with pixel value 0 and free space with pixel value 1 and unknown space with a gray value between 0 and 1. This map is given as an input to program and with the help of ***Algorithm 2***, which replicates the occupancy grid map for implementing path planning using the proposed DT – RRT* for the mobile robot to move freely without collisions. The path is generated and given as an input to the Closed Loop Control as described in ***Figure 4***, and the velocities are published to the ***cmd_vel*** topic. The Robot simulation is done in Gazebo with the help of **ROS (Robot Operating System)**. The Simulation is visualized in ***Figure 10.***

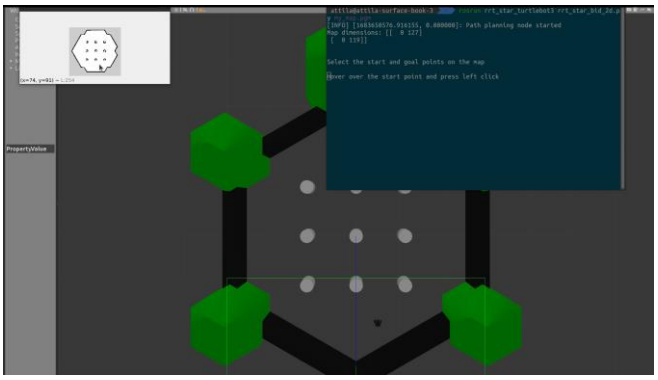

Figure 10: Gazebo Simulation with ROS

VII. Conclusion & future scope

In this paper, we successfully implemented the Double Tree RRT* algorithm with SLAM integration for path planning on the TurtleBot3 Burger robot. By combining these techniques, we achieved improved mapping, localization, and path planning capabilities in complex and unknown environments. The experimental results demonstrated the effectiveness of the proposed approach, as the algorithm accurately navigated the environment and generated efficient paths. The integration of SLAM techniques allowed us to create accurate maps of the unknown environment and estimate the robot's pose within that map. This enhanced the reliability and accuracy of the path planning process. The comparison between the theoretical path and the robot's actual path confirmed the algorithm's ability to navigate the environment accurately. However, there is still scope for improvement in this project. Firstly, further optimization of the Double Tree RRT* algorithm can be explored to enhance convergence time and path quality. Fine-tuning the algorithm parameters and incorporating additional optimization techniques may lead to even more efficient and optimal paths.

Additionally, the SLAM integration can be improved by considering advanced sensor fusion techniques, such as incorporating data from multiple sensors (e.g., cameras, lidar, etc.) to improve mapping and localization accuracy. Advanced filtering and smoothing algorithms can also be explored to enhance the robustness of the SLAM system in dynamic environments.

Furthermore, conducting experiments in real-world environments would provide valuable insights into the algorithm's performance and its applicability to practical scenarios. Real-world experiments would allow us to validate the algorithm's capabilities under more challenging and diverse conditions, such as varying lighting conditions, dynamic obstacles, and complex terrains.

VII. References

[1] A Fast and Efficient Double-tree RRT*-like Sampling-based Planner Applying on Mobile Robotic Vehicles, April 2018.

[2] Intelligent bidirectional rapidly-exploring random trees for optimal planning in complex cluttered environments. Volume 68, June 2015.

[3] An adaptive rapidly-exploring random tree, IEEE/CAA J. Autom. Sinica, vol. 9, no. 2, pp. 283–294, Feb. 2022.

[4] An adaptive rapidly-exploring random tree, IEEE/CAA J. Autom. Sinica, vol. 9, no. 2, pp. 283–294, Feb. 2022.

[5] MRRT: Multiple Rapidly-Exploring Random Trees for Fast Online Replanning in Dynamic Environments. 22 Apr 2021

[6] Informed RRT*: Optimal Sampling-based Path Planning Focused via Direct Sampling of an Admissible Ellipsoidal Heuristic. 28 Nov 2014.

[7] Extending RRT for Robot planning with SLAM. Applied Mechanics and Materials Vol. 151 (2012) pp 493-497, January, 2012.