Implementing RRT-Connect and Goal biased RRT for motion planning and Comparing the performance with Standard RRT.

1st Vamshi Kalavagunta
A. James Clark School of Engineering
University of Maryland
College Park, MD, USA
vamshik@umd.edu

2nd Suryavardhan Reddy Chappidi
A. James Clark School of Engineering
University of Maryland
College Park, MD, USA
chappidi@umd.edu

***Abstract*—A path planning approach using variants of the RRT algorithm for a non-holonomic autonomous robot in an obstacle space. The RRT-Connect and Goal biased RRT algorithms are implemented in 2D map and Gazebo simulation environment and compared their performance with standard RRT. One of the path planning approach RRT-Connect involves constructing two trees, one from the start configuration and the other from the goal configuration, and incrementally growing them until they connect and other path planning approach is Goal Biased RRT which generates a random node towards the goal node rather than at a random point in the configuration space. The resulting path is used to navigate the non-holonomic robot through the obstacle space to the goal region. The simulations of the above algorithms are done to find the feasible paths for the non-holonomic robot in the high-dimensional configuration by minimizing the path length and computation time.**

***Index Terms*—Path Planning, RRT, RRT-connect, Goal biased RRT, Autonomous Robot, Non-holonomic, Obstacle Space, Gazebo, ROS.**

I. Introduction

Path planning for autonomous robots is an essential problem in robotics, particularly in the context of navigation and exploration. In recent years, significant progress has been made in developing path planning algorithms that can generate optimal paths for robots operating in complex environments. One of the major challenges in path planning is to find an efficient and feasible path that avoids obstacles and satisfies constraints such as non-holonomic motion.

Path planning algorithms for autonomous robots can be categorized into two types: Search-based planning and sampling-based planning. The former discretizes the environment and can be resource-intensive and limited by dynamic constraints. The latter is more scalable, considers dynamic constraints directly, and generates continuous feasible solutions, but may have completeness and efficiency trade-offs. Overall, the sampling-based approach is more versatile and can handle complex and dynamic environments.

Our project aims to explore the capabilities of the Rapidly-exploring Random Trees-Connect (RRT-Connect) and goal biased RRT algorithms, which is an extension of the original RRT algorithm designed to provide optimal paths for non-holonomic robots in obstacle spaces. RRT algorithms are known for their ability to quickly explore an environment to find a feasible path. However, traditional RRT algorithms lack the ability to optimize the path for efficiency.

In this project, the RRT-Connect and Goal biased RRT algorithms were used on ROS Turtlebot 3 to navigate in a configuration space consisting of static obstacles. The simulation results demonstrate the effectiveness of the RRT-Connect algorithm and Goal biased RRT in generating collision-free paths for a non-holonomic robot in an obstacle space. The robot successfully navigates through the environment while avoiding obstacles and achieving its goal pose. However, as no PID controller is being used, sometimes the turtlebot does not move in the desired way. The simulation results also show that the RRT-Connect and Goal biased RRT algorithm are efficient in terms of computation time and resource usage, making it a suitable algorithm for real-time applications.

II. Literature review

A. RRT algorithm

RRT algorithm is used for solving path planning problems in robotics. It builds a tree rooted at the start configuration and iteratively adds new nodes towards the goal configuration.

B. RRT-Connect algorithm

The paper [1] by Kuffner and LaValle presents an efficient approach to single-query path planning using the RRT-Connect algorithm. The authors highlight the limitations of existing path planning algorithms and propose a new algorithm that combines the advantages of the RRT algorithm with the concept of bidirectional search. The RRT-Connect algorithm is specifically designed for non-holonomic robots and generates a smooth and efficient path in complex obstacle spaces.

C. Goal biased RRT

The paper[2] by X. Jin, Z. Yan, H. Yang, Q. Wang and G. Yin, presents an goal-biased RRT (Rapidly-exploring Random Tree) path planning approach for autonomous ground vehicles. The proposed method uses a goal-biased strategy to efficiently explore the configuration space and generate feasible paths. The approach is validated using simulations in Gazebo and the results show that the proposed method is effective in finding feasible paths for the autonomous ground vehicle, while minimizing path length and computation time.

III. Methods

A. Why path planing is needed ?

Path planning is a critical task in the field of robotics, as it enables autonomous robots to navigate through environments safely and efficiently. Path planning algorithms provide a means to calculate an optimal path from the robot's current location to a desired destination, while avoiding obstacles and respecting the robot's constraints, such as its movement limitations and sensor capabilities. Without path planning, robots would not be able to operate in dynamic and uncertain environments, limiting their potential for use in a wide range of applications, including search and rescue, agriculture, and manufacturing. Additionally, effective path planning can result in reduced energy consumption, longer battery life, and improved overall performance of the robot.

B. RRT

RRT generates nodes randomly and then connects each node to the closest available node. Each time a node is created, a check must be made that the node lies outside of an obstacle. Furthermore, chaining the node to its closest neighbor must also avoid obstacles. The algorithm ends when a node is generated within the goal region, or a limit to the number of iterations is hit.

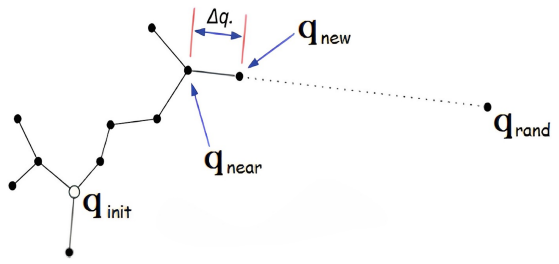

Fig. 1. RRT Growth Expansion.

C. What is RRT-connect ?

The RRT Connect algorithm is a bidirectional approach that grows two random trees from the initial and destination state points to explore the state space. In each iteration, a new random point is sampled and added to the first tree, which then searches for the nearest node in the second tree. The second tree is then expanded towards the nearest node to obtain the next node. This process is repeated until the trees connect or a collision occurs. The algorithm continues to expand the trees until a feasible path is found.

```
RRT(q_start, q_goal)
1    T.add(q_start)
2    q_new ← q_start
3    while(DISTANCE(q_new, q_goal) > d_threshold)
4       q_target = RANDOM_NODE()
5       q_nearest = T.NEAREST_NEIGHBOR(q_target)
6       q_new = EXTEND(q_nearest, q_target, expansion_time)
7       if(q_new != NULL)
8             q_new.setParent(q_nearest)
9             T.add(q_new)
10   ResultingPath ← T.TraceBack(q_new)
11   return ResultingPath
```

Fig. 2. RRT pseudocode.

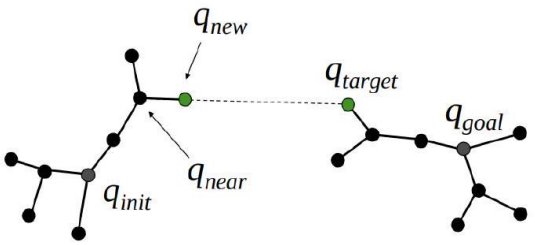

Fig. 3. Bidirectional tree.

```
Algorithm : RRT-connect algorithm pseudo code
V1←{Xinit};E1←φ,G1←(V1,E1);
V2←{Xgoal};E2←φ,G2←(V2,E2);i←i+1;
While i<N do
   Xrand←Sample(i);i←i+1;
   Xnearest←Nearest(G1,Xrand);
   Xnew←Steer(Xnearest,Xrand);
          If obstacleFree (Xnearest,Xnew)then
       V1←V1∪{Xnew};
E1←E1∪{(Xnearest,Xnew)};
Xnearest←Nearest(G2,Xnew);
Xnew←Steer(Xnearest,Xnew);
If  ObastacleFree(X'nearest,X'new)then
  V2←V2∪{Xnew};
E2←E2∪{(X'nearest,X'new)};
do
X"new←Steer(X'new,Xnew)
 If ObastacleFree(X"new,X'new)then
    V2←V2∪{Xnew"}
    E2←E2∪{(Xnew",Xnew')};
    X ' ←Xnew";
 Else break;
    While not Xnew'=Xnew
   If Xnew'=Xnew then return(V1,E1);
    If |V2|<|V1| then Swap(V1,V2);
```

Fig. 4. RRT- connect Pseudocode.

D. Goal biased RRT

Goal-biased Rapidly-exploring Random Trees (RRT) is a variant of the RRT algorithm used for path planning in robotics. This algorithm prioritizes the sampling of nodes in the vicinity of the goal configuration to guide the search process towards the goal. It combines the RRT algorithm's ability to efficiently explore high-dimensional configuration spaces with obstacles with a goal bias, which makes it effective in finding feasible paths for autonomous robots in complex environments. By sampling nodes near the goal region, the algorithm can efficiently explore the space while ensuring that the solution path is optimal and obstacle-free.

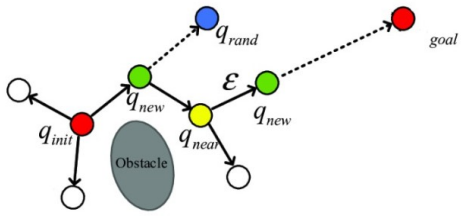

Fig. 5. Goal Biased RRT Expansion.

Algorithm 2: Goal Biasing RRT

```
T ← InitialSetting()
while (norm(q_goal - q_new)≥ε)
  q_rand ← RandomSampling()
  q_rand ← GoalBiasingSampling(q_goal)
  q_nearest ← FindNearest(T, q_rand)
  q_new ← Steer(q_nearest, q_rand, ε)
  if CollisionCheck(q_nearest, q_new)
    T(N,E) ← AddNode(T, q_new)
  end if
end while
N_path, t_cal, C_path ← GetResult(T, q_goal)
return N_path, t_cal, C_path
```

Fig. 6. Goal Biased RRT pseudocode.

IV. Implementation and Results

The RRT, RRT-Connect, and Goal-Biased RRT algorithms were implemented in two distinct environments, low density and high density, for 2D visualization. As shown in the figures below, Map-1 contains only 10 obstacles (15cm cubes) while Map-2 contains 39 obstacles (15cm cubes), with most sides of the goal point covered by obstacles. Map-1 environment was simulated using Gazebo.

We used the ROS Turtlebot robot for our simulations.The goal is to plan a collision-free path for the Turtlebot robot from its starting position to a goal position while avoiding obstacles. The resulting path is to navigate the Turtlebot robot through the obstacle space to the goal region. Our simulations showed that the RRT-Connect algorithm is most effective in finding feasible paths for the non-holonomic robot in the obstacle space, minimizing path length and computation time. We also found that the Goal Biased RRT algorithm was able to efficiently explore the configuration space and generate feasible paths than the standard RRT in the low density maps.

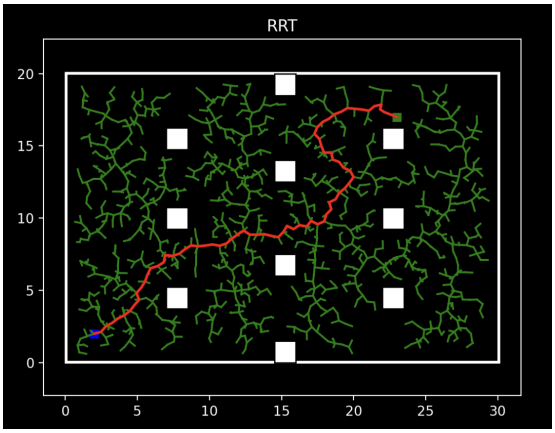

Fig. 7. RRT Map-1.

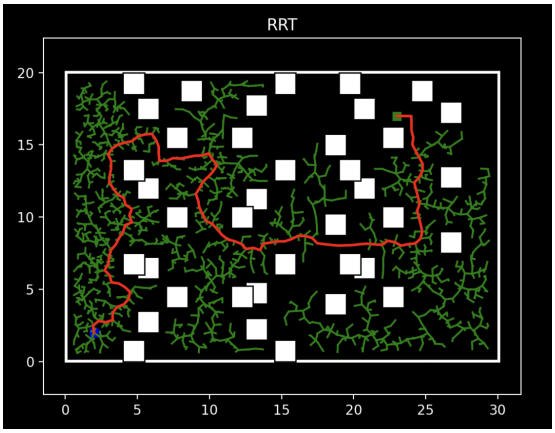

Fig. 8. RRT Map-2.

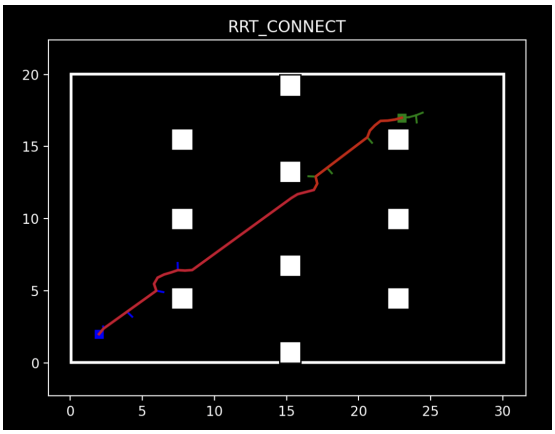

Fig. 9. RRT-Connect Map-1.

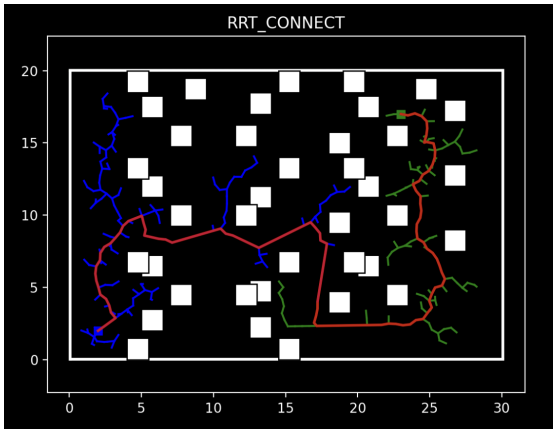

Fig. 10. RRT-Connect Map-2.

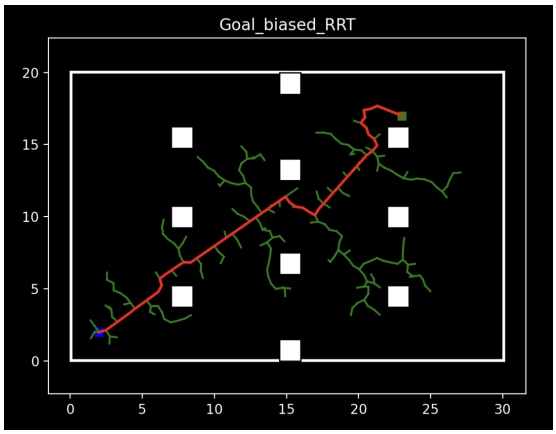

Fig. 11. Goal Biased RRT Map-1.

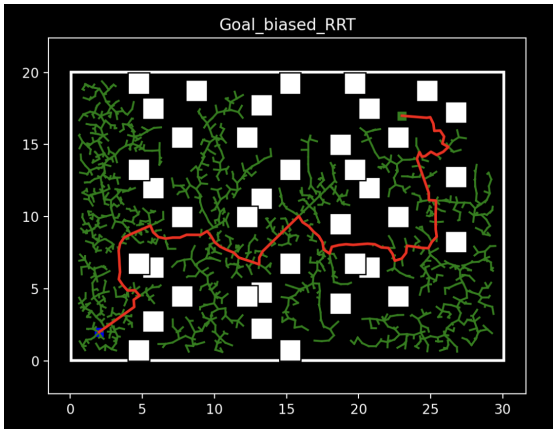

Fig. 12. Goal Biased RRT Map-2.

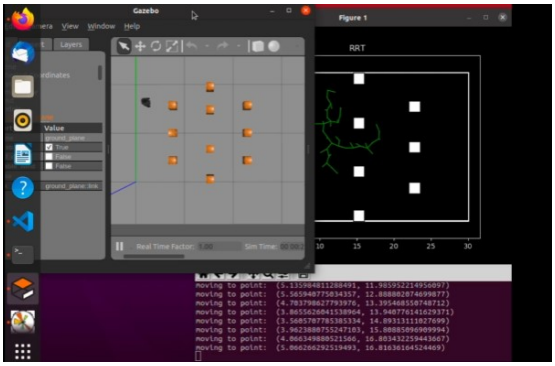

Fig. 13. RRT Gazebo.

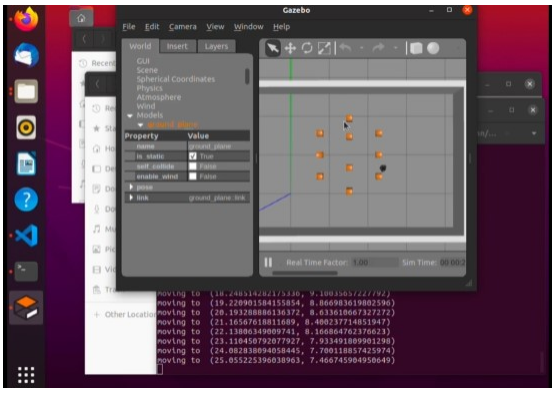

Fig. 14. RRT- Connect Gazebo.

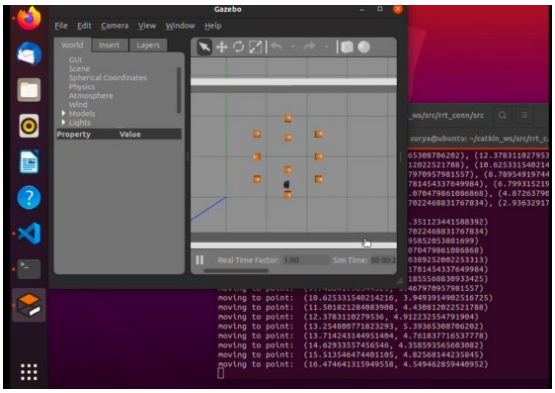

Fig. 15. Goal Biased RRT Gazebo.

V. Simulation Results

There are three sets of results shown, each for a different scenario in Map1 and Map2. For each scenario, the table shows the total time taken by each algorithm to find a path, the total number of iterations performed, the total number of nodes generated in the search tree, and the number of nodes in the optimal path found by each algorithm.

Based on the tables below, it appears that the RRT-Connect algorithm performs the best overall in terms of total time taken and nodes in the optimal path, while also generating the fewest number of nodes in the search tree. The Goal-Biased RRT algorithm also performs well in Map-1, but it takes more time in Map-2 as there are more obstacles surrounding the goal, leading to a higher number of iterations and a smaller search space. In this case, the standard RRT algorithm outperformed the Goal-Biased RRT algorithm in terms of time taken and iterations.

TABLE I
Results for Map1

	RRT	**Goal Biased RRT**	**RRT-Connect**
Total Time	0.983	0.196	0.047
Total Iterations	1799	1660	74
Total Nodes	1440	241	70
Nodes in Optimal Path	73	59	57
Total Time	0.592	0.775	0.050
Total Iterations	1183	4381	75
Total Nodes	923	779	80
Nodes in Optimal Path	72	62	59
Total Time	0.367	0.204	0.054
Total Iterations	850	1524	89
Total Nodes	592	264	69
Nodes in Optimal Path	74	63	57

TABLE II
Results for Map2

	RRT	**Goal Biased RRT**	**RRT-Connect**
Total Time	2.846	6.797	0.908
Total Iterations	4221	23456	1785
Total Nodes	1107	1704	392
Nodes in Optimal Path	112	105	121
Total Time	5.282	9.019	0.897
Total Iterations	6085	28608	2042
Total Nodes	1894	2050	406
Nodes in Optimal Path	115	127	97
Total Time	4.662	11.349	0.711
Total Iterations	5027	35719	2147
Total Nodes	1844	2392	296
Nodes in Optimal Path	128	126	94

VI. Conclusion

Our simulations showed that both RRT-Connect and Goal Biased RRT algorithms were able to generate collision-free paths for the Turtlebot robot in an obstacle space. The resulting paths were smooth and efficient, and the robot was able to navigate through the environment while avoiding obstacles and achieving its goal position.

We found that the RRT-Connect algorithm was particularly effective in minimizing path length and computation time, making it a suitable algorithm for real-time applications. The Goal Biased RRT algorithm was also efficient in terms of computation time and was able to generate feasible paths in a shorter time than the standard RRT algorithm.

References

[1] J.J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065), 2000, pp. 995-1001 vol.2, doi: 10.1109/ROBOT.2000.844730.doi:10.1109/CCDC52312.2021.9602848.

[2] X. Jin, Z. Yan, H. Yang, Q. Wang and G. Yin, "A Goal-Biased RRT Path Planning Approach for Autonomous Ground Vehicle," 2020 4th CAA International Conference on Vehicular Control and Intelligence (CVCI), Hangzhou, China, 2020, pp. 743-746, doi: 10.1109/CVCI51460.2020.9338597.

[3] C. Zhao, X. Ma and C. Mu, "Research on Path Planning of Robot Arm Based on RRT-connect Algorithm," 2021 33rd Chinese Control and Decision Conference (CCDC), 2021, pp. 3800-3805, .