Lab 2 Report - Planning and Navigation

Team: Aditya Jain, Geet Patel, Sugumar Prabhakaran

1 Introduction

Lab 2 focused on planning and navigation using perfect localization for the Turtlebot 3 Waffle Pi in a simulation. The first objective was to build a global planner using RRT and RRT* that finds a path from a start point to a goal location in a known environment (i.e. known map). The second objective was to follow the generated path from the global planner using a local planner and controller to avoid obstacles in real-time.

2 Deliverables

2.1 RRT

Figure 1 illustrates the path generated using RRT where the start position is in the top-left and the goal position is in green at the bottom-right. The tree generated by RRT is in blue whereas the red line indicates the path from start to goal location. In our RRT implementation, the points are randomly sampled within the bounds of the map and every 7$^{\text{th}}$ sample is the goal location itself. This ensured that the tree was growing towards the goal while exploring adequately. For every sample goal point, trajectory rollout is done using the robot kinematics for 5 seconds or till the sample goal point is reached, whichever occurs first.

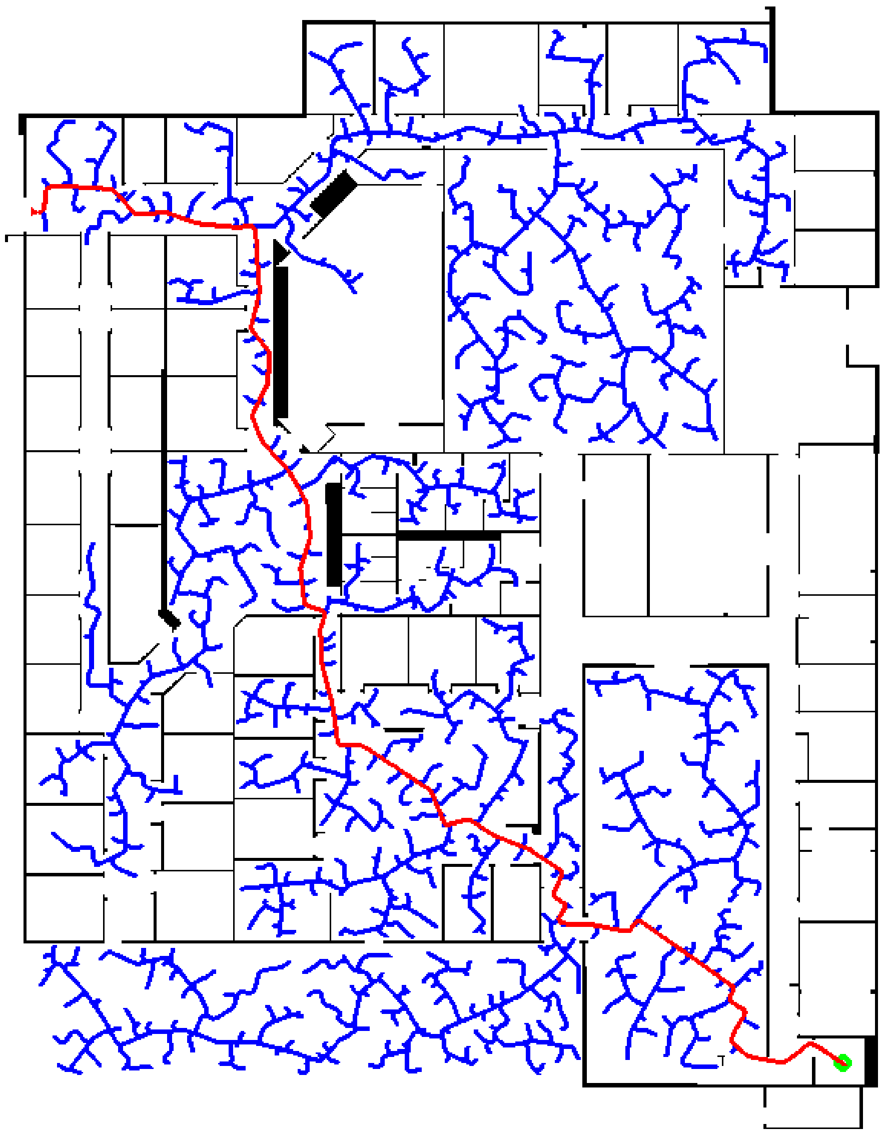

Figure 1: Path Generated from RRT

2.2 RRT*

Figure 2 depicts the result from the RRT* algorithm using the same color scale. RRT* builds over our implementation of RRT. After a point is sampled and connected to the nearest node, two kinds of re-wiring are performed. First, in a vicinity of 0.4 m of the last added node that we name as *min_node*, if the cost-to-come from any other node is less than the current cost-to-come of the *min_node* and path is collision-free, the *min_node* is re-wired to the neighbour node. Second, all the nodes in the vicinity are checked for cost-to-come with the *min_node*. If the cost-to-come from *min_node* is lower than their current cost-to-come and have a collision-free path, then the node is re-wired to *min_node* and the cost difference is propagated to all the child nodes. Now this node becomes the *min_node* and the process continues for all the remaining points in the vicinity. It can been seen in figure 2 that the path from RRT* is smoother than RRT.

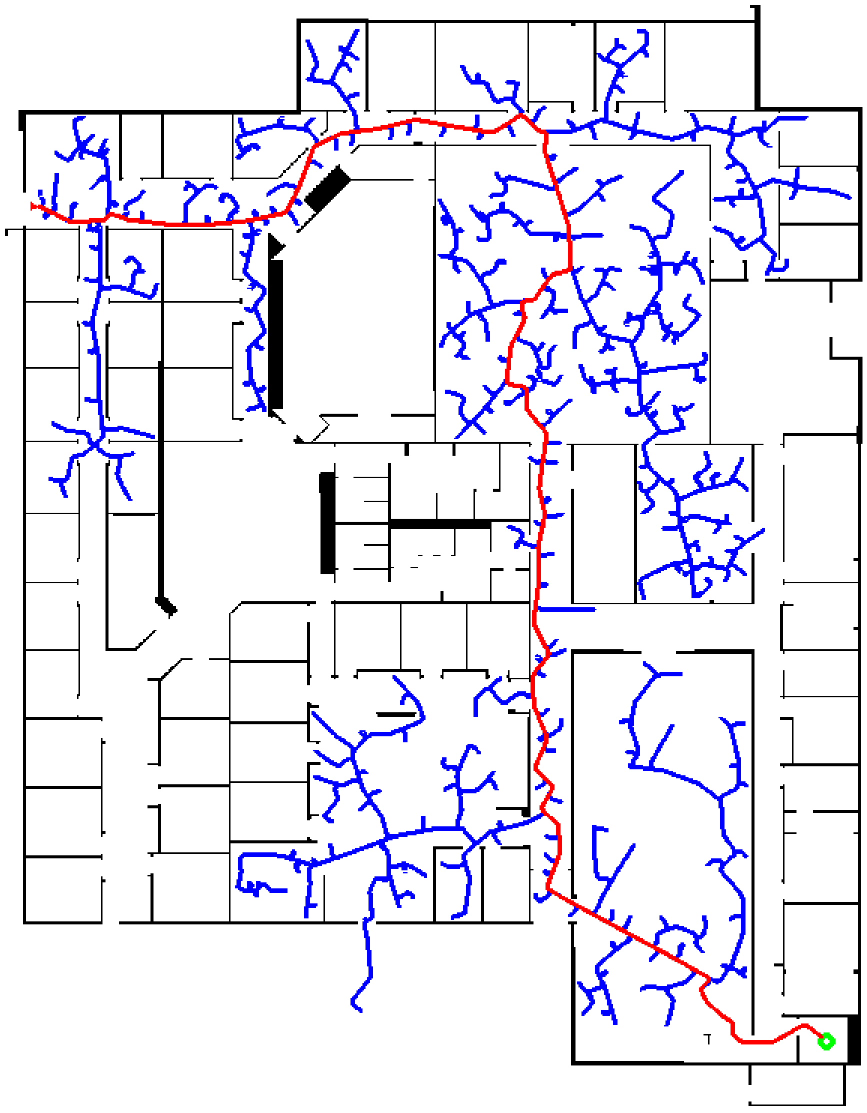
Figure 2: Path Generated from RRT*

2.3 Trajectory Rollout in Simulation

A simulation of our local planner using trajectory rollout to follow our RRT-generated path is shown in the attached video. This was implemented in the `l2_follow_path.py` in three steps:

- **Step 1 - Simulate N Trajectories:** Using 4 velocities and 11 angular velocities, N=44 total combinations of command inputs were generated. For each of these options, the simulated robot poses over the next 5 seconds split into 200 time-steps was calculated using the standard unicycle model from Lesson 5 - Vehicle Model notes.

- **Step 2 - Check Trajectories for Collisions:** Each of the 44 trajectory options were next checked for collisions. To improve performance, we limited the collision checking to a generated sample space of 20 x 20 pixels around the robot's current position. We looped through each time-step of every option to find the lowest distance to collision for each option. If the lowest distance to an object was less than the threshold value of 0.225 m, the trajectory option was removed from valid trajectory options for the current control loop.
- **Step 3 - Calculate Optimal Trajectory:** In order to determine which of the remaining trajectories was optimal, we used a cost function approach. A quadratic cost function of euclidean distance between the milestone pose and the predicted pose at 75th time-step of each option and the option with the least cost was chosen as the input. We chose the 75th time-step as it gave smoother control while progressing well towards a milestone. Further, we only included the orientation of the robot in the cost function once the robot was sufficiently close to a milestone position. This is because some of the orientation requirements for the milestones, were fairly different than the intuitively required orientation for the robot to reach that milestone.