Learning cost-to-go functions to guide RRT* algorithm for manipulator

Pranav Patil, Rucha Deshpande

***Abstract*—Motion Planning Algorithms need to be adaptive to complex environments for practical applications. RRT* is a conventional sampling-based algorithm that attempts to find an optimal obstacle-free path. However, it faces challenges as the complexity of the planning problem rises. The quality of returned solutions and runtime is affected in more complex environments. In this project, we attempt to modify RRT* algorithm by using a neural network to predict cost-to-go and provide better solutions in a fair amount of time.**

***Index Terms*—Path planning, Robots, Motion planning**

I. Introduction

MOTION planning is a crucial aspect of practical robotics applications. Conventional algorithms like Rapidly Exploring Random Trees achieve good results under specified conditions. However, these algorithms tend to become less efficient in terms of convergence time and quality of the solution with increasing environment complexity. In RRT* algorithm, the role of rewire function is to check if the newly added node to the path serves as a better parent to already existing near nodes. As the number of nodes in the tree increase, checking the cost from start to intermediate node becomes more and more tedious. In this project, we address this issue by building a modified RRT* motion planner that aims to reduce the runtime of the algorithm in complex environments while giving efficient solutions. We use a neural network to predict the cost to goal from intermediate nodes as a replacement to the rewire function in RRT* and boost the process of optimizing the path from start to goal.

II. Background

A. RRT* - Rapidly Exploring Random Trees

Rapidly Exploring Random Trees [4] are a sampling-based method to compute efficient motion plans. An RRT grows a tree rooted at the starting configuration by using random samples from the search space [2]. A connection is attempted between each drawn sample it and the near nodes in the tree. If a collision-free path connecting the near nodes and the sampled node is feasible, the sampled node is added to the tree. For each connected node, the 'rewire' function is performed which checks if the cost to the nodes in the nearest neighbors is less through the newly added node as compared to their older costs. If less cost path is found, the near neighbor's parent is changed to new node. The RRT* algorithm is as follows:

P. Patil and R. Deshpande were with the Department of Computer Science, Purdue University, West Lafayette, IN, 47906 USA e-mail: patil127@purdue.edu, deshpa37@purdue.edu.

B. PRM - Probabilistic Road Maps

The probabilistic roadmap, PRM [3] is another sampling-based motion planning algorithm, which solves the problem of determining a path between a starting configuration of the robot and a goal configuration while avoiding collisions. The algorithm runs in two phases, a learning phase and a query phase. In the learning phase, a probabilistic roadmap is constructed and stored as a graph. For creating this graph, random samples are taken from the environment and tested if they lie in the free space. If they do, near nodes within a fixed radius of the sampled nodes are found. The sampled nodes are then connected to the graph if there exists a collision-free path joining them with the near nodes. In the query phase, any given start and goal configurations of the robot are connected to two nodes of the roadmap; the roadmap is then searched for a path joining these two nodes. In our implementation, we use a modified version of PRM i.e PRM* which uses a variable radius for finding nearest neighbors. The radius is determined based on the number of nodes and the number of dimensions.

III. Implementation

In milestone 1, we have set up 2D and 3D environments for motion planning. We have implemented the PRM* algorithm to obtain the network graph of possible paths. We have also implemented the shortest path algorithm that gives the least cost path between randomly sampled start and goal nodes in the environment. The details of environment and PRM* algorithm are as follows:

A. Environment Setup

1) 2D Environments (Point Robot):

- Number of obstacles = 7
- Sizes of obstacles = Variable

2) 3D Environments: The 3D environment is setup in pybullet with a 3-DOF UR5 manipulator.

- Number of obstacles = 7
- Sizes of obstacles = $0.2 \times 0.2 \times 0.2$

B. PRM* Implementation

We create the network graph of 4000 nodes in 2D and 3D environment. The start and goal node are sampled randomly and the shortest path is computed using Dijkstra's Shortest Path Algorithm [1].

The algorithm of PRM* is as follows:

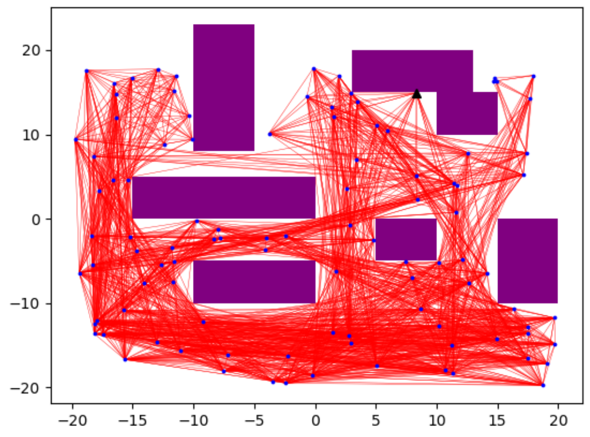

Fig. 1: PRM* Graph for 2d Environment (100 Nodes)

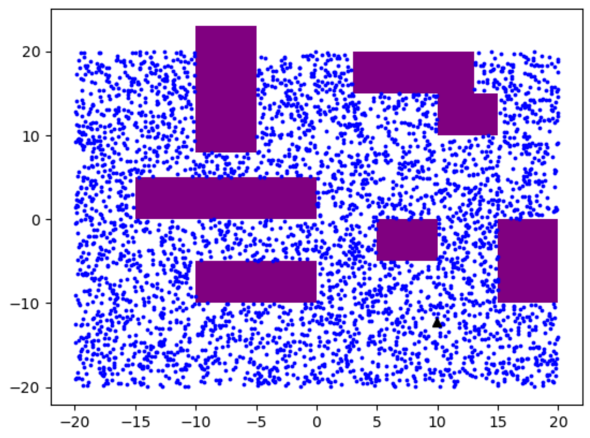

Fig. 2: PRM* Nodes for 2d Environment (3600 Nodes)

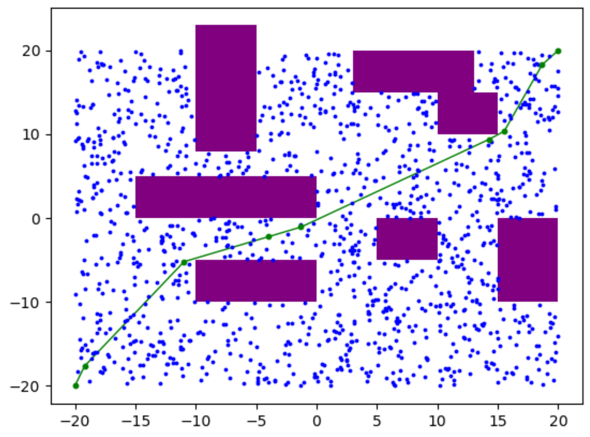

Fig. 3: Shortest path from a start and goal configuration (two corners of environment in this case, cost = 58.96)

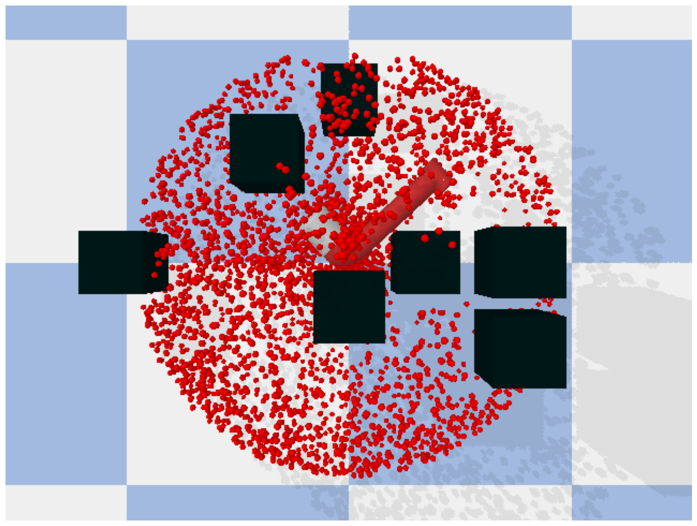

Fig. 4: 3D Environment 3400 nodes (top view)

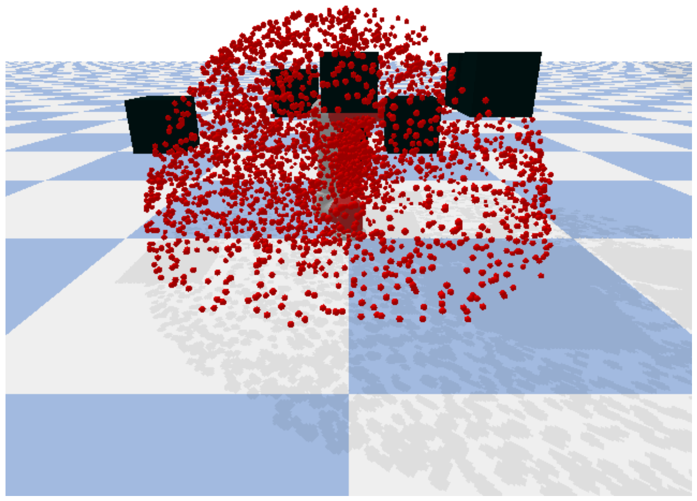

Fig. 5: 3D Environment 3400 nodes (side view)

```
Initialize $G = (V, E)$
**for** i=0 to N **do**
  $q_{rand} \leftarrow sample(i)$
  $N_q \leftarrow select\_nearest(q_{rand}, G)$
  **for** all $q' \in N_q$ **do**
    **if** $(q_{rand}, q') \in E$ && $steerA(q_{rand}, q')$ **then**
      $E \leftarrow E \cup \{q_{rand}, q'\}$
    **end if**
  **end for**
**end for**
```

For finding the nearest neighbors, we consider the nodes that are within a particular radius r of the currently sampled node. The radius is computed by the following formula:

$$r = \gamma * \Big(\frac{log(n)}{n}\Big)^{\frac{1}{d}}$$

where, γ is a constant, is number of nodes and d is degree of freedom. $steerA$ is the function to check if there is a direct

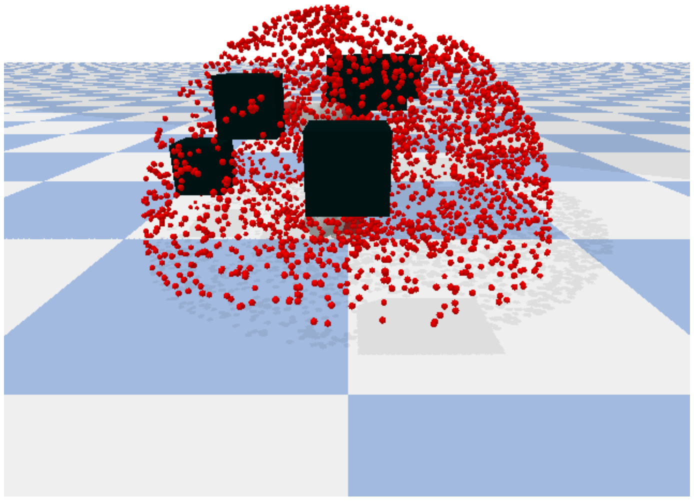
Fig. 6: 3D Environment 3400 nodes (front view)

collision free path between two points.

IV. Conclusion

In milestone 1, we completed the environment setup and implemented PRM* to generate the network of paths. We sampled random start and goal nodes and found the shortest paths and their cost using PRM*. We are currently working on building a dataset of shortest paths and their costs obtained from the generated graph networks of different environments. We will use this dataset to train a neural network for predicting the cost from a given node to the goal. For milestone 2, we will attempt to complete the training of the network and use it as a replacement for the rewire function in RRT*.

References

[1] Dijkstra's algorithm. https://en.wikipedia.org/wiki/Dijkstra

[2] Rapidly-exploring random tree. https://en.wikipedia.org/wiki/Rapidly-exploring$_random_tree$.

[3] L.E. Kavraki, P. Svestka, J.-C. Latombe, and M.H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, 1996.

[4] Steven M. LaValle. Rapidly-exploring random trees : a new tool for path planning. *The annual research report*, 1998.