RBE 550 Standard Search Algorithms Implementation Documentation

Alex Tacescu — Spring 2021

I. Introduction

This document accompanies my homework 2 submission. It is designed to explain my thought process when developing my code. It will also explain the implementation of two different algorithms: Probabilistic RoadMaps (PRM) and Rapidly-Exploring Random Tree (RRT). PRM will be implemented with 4 different sampling methods, while RRT will also be implemented with another variant (RRT*).

II. Questions

A. For PRM, what are the advantages and disadvantages of the four sampling methods in comparison to each other?

Uniform sampling guarantees uniform distribution between points. However, without a lot of points, uniform sampling can fail to find an optimal solution when there are narrow passage-ways without many iterations.

Random sampling similarly attempts to give all locations equal opportunity to become a node, while also adding some randomness to attempt to curtain the issues with Uniform sampling. However, it too can fail to find a solution when narrow and curvy passageways exist between the start and end goal. It also can occasionally fail to sample all areas due to it's random selection process, especially if there are a low number of iterations.

Gaussian Sampling attempts to ”look” for obstacles, and will generally pick nodes close to the edges of obstacles. This is useful to try and identify obstacles, but it can sometimes lead to a failure to find a good solution, especially when there is little to no clutter in the environment

Bridge Sampling is very effective at finding points when there are narrow passage ways. It is however slower than other sampling methods, and suffers from the same issues that Gaussian sampling does, where a lot of points are required and finding a good solution in low-clutter environments is not guaranteed.

B. For RRT, what is the main difference between RRT and RRT? What change does it make in terms of the efficiency of the algorithms and optimality of the search result?*

RRT* is a more optimized RRT. Simply put, RRT* can lead to a shorter path in less iterations. This is due to RRT*'s capability to 'rewire' itself, where existing nodes look around for shorter paths. Since RRT* is an anytime algorithm, it can also come up with many different paths, and pick between the best ones. While RRT usually can create really wavy paths, RRT* generally creates very straight and direct paths to the goal. It is also important to note that RRT* usually much slower per iteration, since it needs to rewire all past nodes.

C. Comparing between PRM and RRT, what are the advantages and disadvantages?

PRM is a great algorithm when multiple queries are needed. It is also probabilistically complete as an algorithm. However, guaranteeing a solution can be hard, especially with small passageways.

RRT and RRT* are very powerful for high degree-of-freedom collision avoidance. They are also good at exploring unexplored space due to its biases. RRT* also can guarantee optimality due to its anytime nature. However, they are both single-query search algorithms, which often make them very expensive for multi-search situations. RRT is also usually suboptimal.

III. Results & Explanation

A. Uniform Sampling PRM

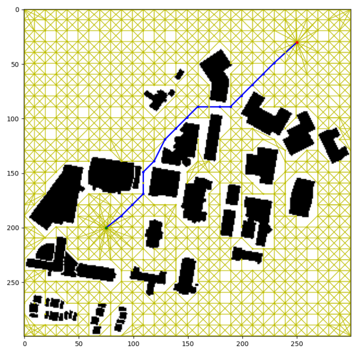

Fig. 1. Uniform Sampling PRM run with 1000 iterations. The graph had 815 nodes and 2971 edges. The length of the path was 260

B. Random Sampling PRM

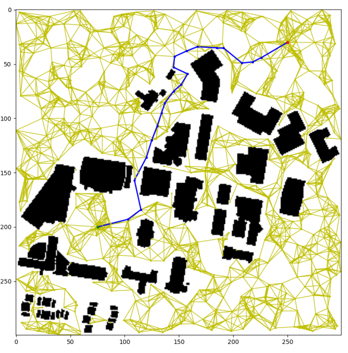

Fig. 2. Random Sampling PRM run with 1000 iterations. The graph had 831 nodes and 3180 edges. The length of the path was 322

C. Gaussian Sampling PRM

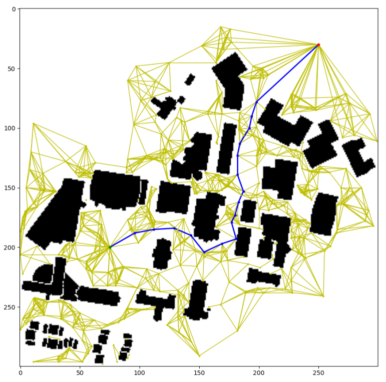

Fig. 3. Gaussian Sampling PRM run with 2000 iterations. The graph had 425 nodes and 1474 edges. The length of the path was 312

D. Bridge Sampling PRM

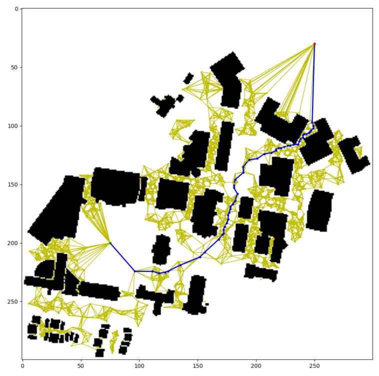

Fig. 4. Bridge Sampling PRM run with 2000 iterations. The graph had 1974 nodes and 7647 edges. The length of the path was 330

E. RRT

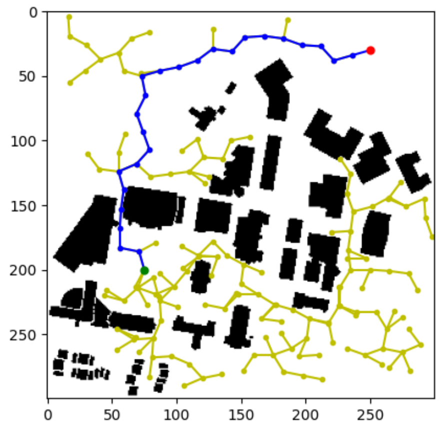

Fig. 5. RRT run with 2000 iterations. It took 161 nodes to find a path length of 374

F. RRT*

I did not get a chance to finish my implementation of RRT*. I was having some issues with my rewire function, although it should be the only thing remaining before RRT* implementation works.

G. Additional Notes and resources

I used a few resources. They are cited below:
Collision checking algorithms: [5] [2]
RRT and RRT* Resources: [3] [4] [6]
Some additional resources I used: [1] [7]

References

[1] [Online]. Available: https://numpy.org/doc/stable/reference/random/generated/numpy.random.uniform.html.

[2] J. Bialkowski, S. Karaman, M. Otte, and E. Frazzoli, "Efficient collision checking in sampling-based motion planning," in *Algorithmic Foundations of Robotics X*, Springer, 2013, pp. 365–380.

[3] T. Chin, *Robotic path planning: Rrt and rrt**, Feb. 2019. [Online]. Available: https://theclassytim.medium.com/robotic-path-planning-rrt-and-rrt-212319121378.

[4] I. Noreen, A. Khan, and Z. Habib, "A comparison of rrt, rrt* and rrt*-smart path planning algorithms," *International Journal of Computer Science and Network Security (IJCSNS)*, vol. 16, no. 10, p. 20, 2016.

[5] J. Pan and D. Manocha, "Fast probabilistic collision checking for sampling-based motion planning using locality-sensitive hashing," *The International Journal of Robotics Research*, vol. 35, May 2016. DOI: 10.1177/0278364916640908.

[6] *Rapidly-exploring random tree*, Feb. 2021. [Online]. Available: https://en.wikipedia.org/wiki/Rapidly-exploring_random_tree.

[7] *Scipy.spatial.kdtree*. [Online]. Available: https://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.KDTree.html.