Hushmand Esmaeili
RBE 550 Motion Planning
RRT and RRT* Algorithm Implementation

Code Algorithm and Implementation

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?

The main difference between RRT and RRT* is that in RRT*, when a new node is added, the new node is rewired as well as all the neighboring nodes. In RRT, however, the new (extended) node is automatically added to the tree if there is no collision.

In terms of the efficiency of the algorithms, RRT* consistently takes longer to complete than RRT. The rewire function decreases the efficiency of RRT* in comparison to RRT. On average, it more or less takes RRT* 10-15 seconds more time to complete than RRT. RRT* also took about 864 nodes (out of an attempt of 1000) on average to find a path, while it only took RRT about 164 nodes (out of an attempt of 1000) to find a path. Clearly, RRT was more efficient.

In terms of optimality, RRT* consistently found the more optimal (shorter) path to the goal in contrast to RRT. RRT* found paths to the goal of average length of 253, while RRT found paths to the goal of average length of 337. This makes sense since RRT* is an optimized version of RRT. In theory, when the number of nodes approaches infinity, RRT* will find the shortest path to the goal.

Note: The numbers presented are based on 3 trials. Screenshots of the results can be found in the Results section.

Compare RRT with the results obtained with PRM in the previous assignment. What are the advantages and disadvantages?

RRT is much faster than PRM. This may be due to the number of edges that need to be calculated is lower for RRT. One advantage of PRM over RRT is that PRM is multi-query, while RRT is single-query, since the tree expands from Start to Goal. PRM's multi-query feature is great for static environments, like the ones we're simulating in this assignment. However, for dynamic environments, PRM cannot be used as multi-query, and hence RRT is better since it is quicker to compute.

Hushmand Esmaeili
RBE 550 Motion Planning
RRT and RRT* Algorithm Implementation

Test Example, Test Result and Explanation

Efficiency (number of nodes) and optimality (path lengths) results for RRT and RRT*

Trial 1:

```
It took 225 nodes to find the current path
The path length is 322.59
It took 863 nodes to find the current path
The path length is 263.93
```

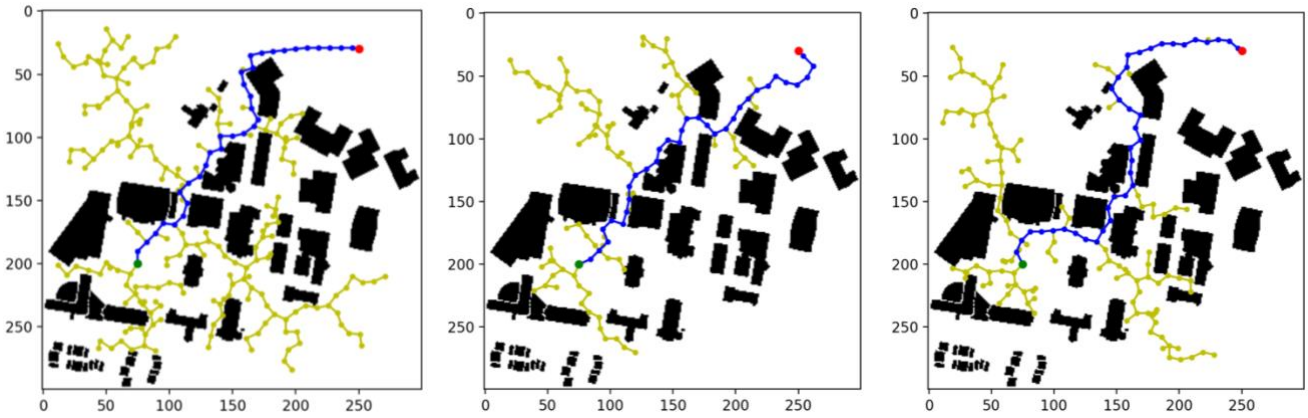
Trial 2:

```
It took 117 nodes to find the current path
The path length is 323.24
It took 859 nodes to find the current path
The path length is 270.07
```

Trial 3:

```
It took 152 nodes to find the current path
The path length is 367.36
It took 870 nodes to find the current path
The path length is 226.78
```

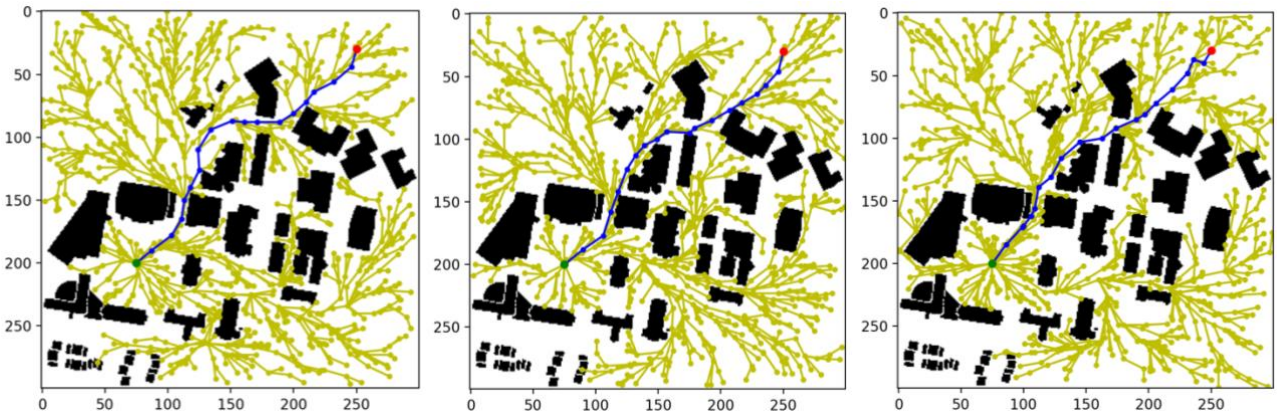
Results for 3 Trials of RRT



The results for RRT above show the efficient (not that many nodes) but less optimal (longer paths) solution to the path planning problem with constant Start and Goal nodes in comparison to RRT*.

Hushmand Esmaeili
RBE 550 Motion Planning
RRT and RRT* Algorithm Implementation

Results for 3 Trials of RRT*



The results for RRT* above show the optimal (short paths) but less efficient (many more nodes) solution to the path planning problem with constant Start and Goal nodes in comparison to RRT.

One interesting observation about the tree shape of RRT* is that it is visually different to that of RRT. The shape of the branches of RRT* looks like more branches branch out from common nodes, and it looks more “straight” and less “spread out”. RRT looks pretty spread out.

Hushmand Esmaeili
RBE 550 Motion Planning
RRT and RRT* Algorithm Implementation

Resources

- Lecture Slides for RBE 550 Motion Planning: Sample-based planning
- https://github.com/motion-planning/rrt-algorithms
- https://courses.cs.washington.edu/courses/cse571/21sp/homeworks/hw3.pdf