Peter Subacz – RBE550 – Path Planning, Coding Assignment 3

Rapidly Exploring Random Trees

Rapidly Exploring Random Trees (RRT) is an incremental sample-based motion planner. The RRT algorithm works by incrementally sampling the configuration space and reducing the distance to each configuration space. RRTs are highly effective at higher dimensional path planning problems that involve obstacles and differential constraints such as nonholonomic constraints [1]. RRTs grow trees as seen in the flow diagram of figure 1. Each tree is grown outward from a root node in towards a random point. If the point is outside of the incremental movement, the nearest direction of nearest node to the point is calculated using a heuristic, such as the Euclidian distance, and used as the nearest node to the point p. This process is iteratively computed until a final solution is generated. It should be noted that naïve RRT algorithms generate sub-optimal solutions without post processing of an RRT.

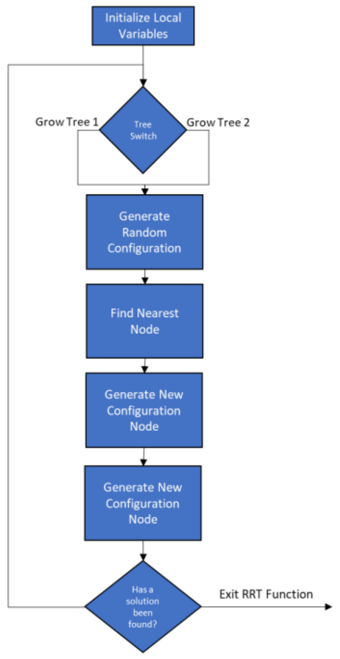
Initialize Local Variables
Grow Tree 1
Tree Switch
Grow Tree 2
Generate Random Configuration
Find Nearest Node
Generate New Configuration Node
Generate New Configuration Node
Has a solution been found?
Exit RRT Function

Figure 1- RRT Flow Diagram

In this Assignment, an RRT algorithm was implement flowing the pseudocode in [1]. And then duplicated with a switch for bidirectional RRT. The main RRT function is described in 3 main blocks:

1. Generate Random Configuration

   In this step a random point in the configuration space is generated. This can be done via naive random point generation, voronoi graph point generation as shown in [2], or a variety of other methods. The key to this step is to provide a point (an y-x coordinate pair) inside the navigable configuration space.

2. Nearest Neighbor Search

In this step a tree graph is searched for proximity to the randomly generated point. The search algorithm used can vary based on the implementation. In this implementation the smallest Euclidian distance to the point is used due to the simplicity of the configuration spaces. In higher dimensional space, search algorithms such as K-Nearest Neighbors, Support Vector Machines, or machine learning models would be better suited to search for the nearest neighbor.

3. New Configuration Node

This step establishes create a new node and inspects the node for a line of sight to the previous node new node. Once a new node has been deemed valid, it is added to a respective tree.

The Figures below show a implementation of a RRT Algorithm which search a few maps. These maps highly the sub optimal solution of a generic RRT algorithm. One interesting characteristic of this algorithm is its implementation is initially biased. The RRT algorithm can be improved to cover via adding node heuristics. Each which a new node is searched for in step 2 a sub step can be called which calculates the cost to move from point to point within an incremental step [3]. Many different cost functions can exist which will influence the shape of a tree. Additionally, In step 1, the points can be assigned by biasing the random point generation towards unexplored areas. This allows a the tree to cover more area

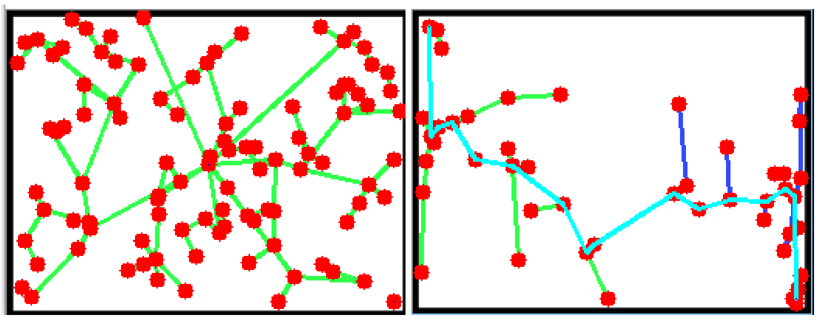

Figure 2 - Blank Map Search - The left side of each figure shows a generic search using the RRT algorithm while the right figure shows a RRT search*

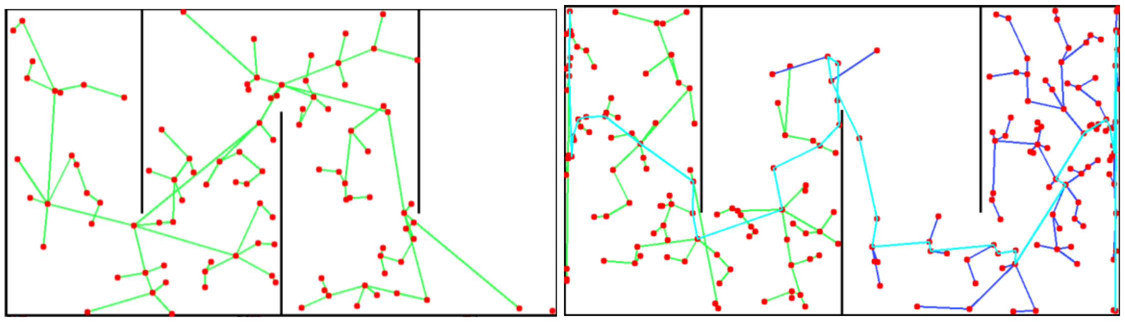

Figure 3- Simple Maze Search - The left side of each figure shows a generic search using the RRT algorithm while the right figure shows a RRT search*

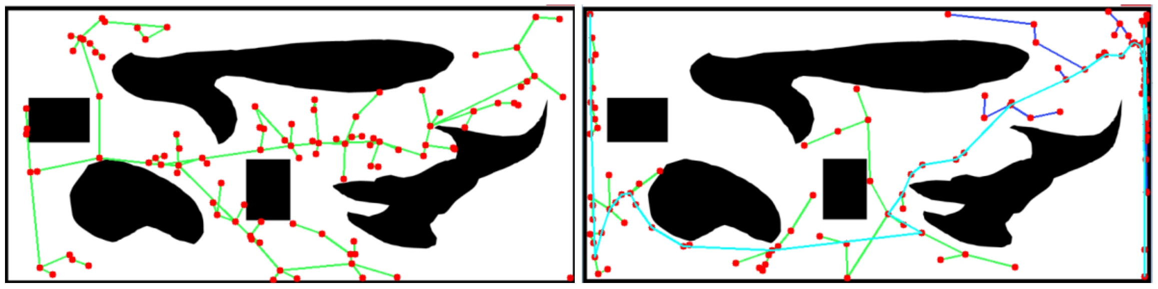

Figure 4 - Simple Hard Search - The left side of each figure shows a generic search using the RRT algorithm while the right figure shows an RRT search*

RRTs can be improved by

References

[1] http://msl.cs.uiuc.edu/rrt/

[2] Huh, Jinwook & Arslan, Omur & D. Lee, Daniel. (2019). Probabilistically Safe Corridors to Guide Sampling-Based Motion Planning.

[3] Karaman, Sertac & Frazzoli, Emilio. (2010). Incremental Sampling-based Algorithms for Optimal Motion Planning.