Planning for High DOF Planar Arm

Planner	Avg. Num of Steps	Avg. Time	Avg. Cost	Avg. No. of Nodes
RRT	21	6.914296981	12.2406608	93
RRT Connect	24	0.046	11.35	65
RRT Star	11	0.15	10.25	387
PRM	8	6.5	12.54	4579

Start and End Configurations Used:

1. ["./map2.txt", "1.715043, 0.760686, 1.567393, 2.612641, 0.331542",
   "0.896447, 2.717722, 0.362222, 4.715409, 1.9524345"],

2. ["./map2.txt", "1.665762, 2.122711, 2.159336, 5.825278, 0.346115",
   "0.442375, 2.669400, 1.684554, 3.232004, 2.894518"],

3. ["./map2.txt", "1.426454, 0.953590, 0.357237, 3.524812, 0.743812",
   "1.010924, 2.990779, 0.700475, 3.774283, 0.197890"],

4. ["./map2.txt", "1.509483, 2.712544, 5.506200, 1.697402, 0.324063",
   "0.316585, 2.262115, 1.549368, 6.278967, 2.685507"],

5. ["./map2.txt", "1.510058, 1.380625, 2.230860, 1.150521, 0.835278",
   "1.691074, 2.297973, 5.233143, 1.019553, 2.999407"],

The above average statistics are on very simple examples and therefore I believe that they do not really represent the whole trend.

Some observations:

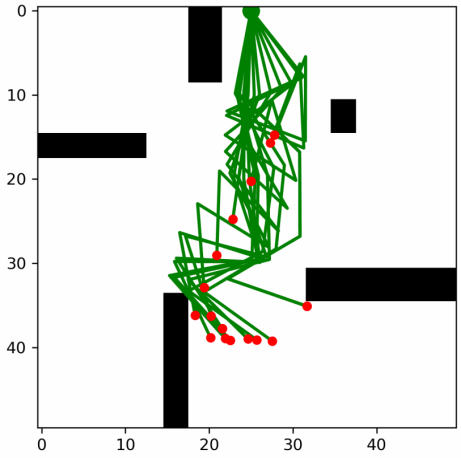
1. RRT and RRT* fail to converge quickly when the goal and start positions are far apart.
2. When the epsilon is high, the path is found quickly.
3. In the given environment, I believe that RRT Connect is the best option. Since we start from both sides, we quickly get the solution and the solution looks reasonable as well. RRT Connect is also the fastest.
4. Issues and improvements with my Planner:
    a. Data structures are not optimized.
    b. Biasing is uniform right now. It can be modified.
    c. Hyperparameters are not properly tuned. There is a chance that the planner fails to find the solution with the current hyperparameters.
    d. Self-collision should be considered.

e. As of now, I am using a Constant radius (epsilon) in RRT* for considering neighbors.
f. Deleting any dynamically created nodes to make the code faster.
g. Right now, I have written Separate codes for everything due to time limitations. Everything can be combined into 1 big class.
h. Can run RRT* for a few thousand extra iterations to make it better.

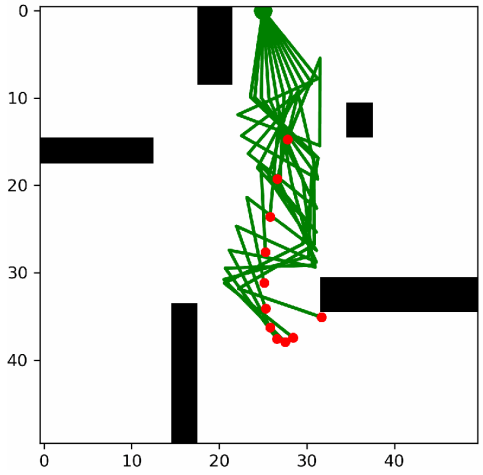
Hyperparameters used:
Epsilon = Pi/4
Goal biasing = 10%
Number of interpolations = 10
If the epsilon is low (0.1) or number of interpolations is high, the planner seems to work best and has high accuracy. But due to machine limits, I am keeping high epsilon so that I do not have to explor more nodes.

I am including .cpp files directly. Therefore normal compilation will compile the code.

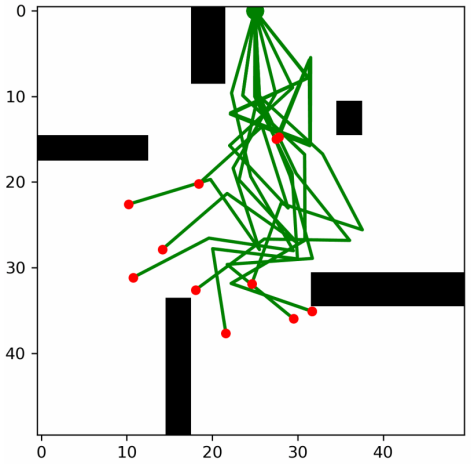
Example results:



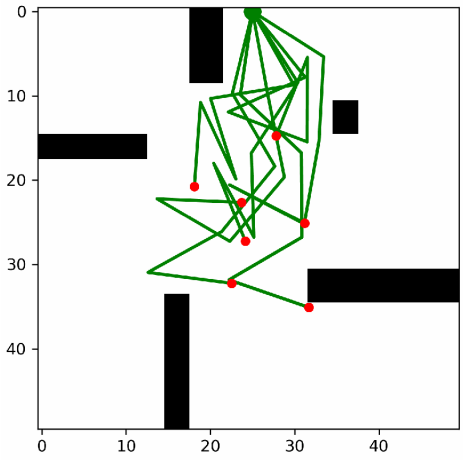
RRT:Example 1



RRT Connect:Example 1



RRT*



PRM