Name: Amani Jammoul
McGill ID: 260381641
COMP 417 - Introduction to Robotics & Intelligent Systems

Assignment 1 RRT

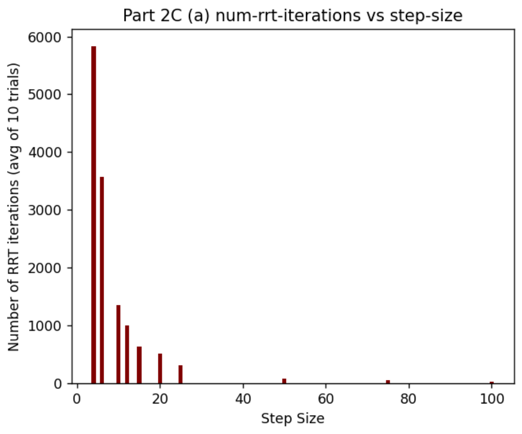
Note: All trials run on “shot.png” world with Gaussian distribution

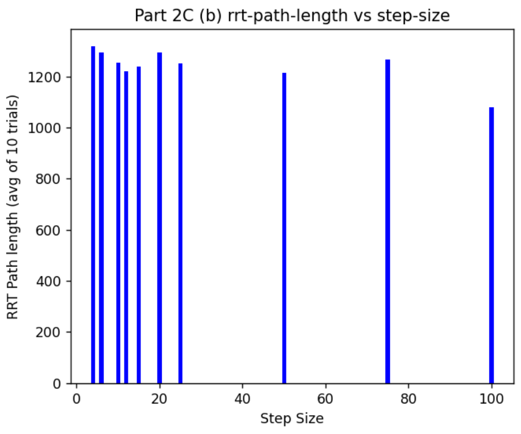
Simple Point Robot

For this first part, I implemented the Rapidly-expanding Random Tree (RRT) algorithm in python for an omnidirectional point robot. I expanded this algorithm for 2 different distributions: uniform and Gaussian (centered on the target point). The guiding point is generated randomly using the python random library, where different seeds can be defined to alter the point generation and path.

While experimenting with the two different distributions, it was clear that the number of RRT iterations was lower overall when using Gaussian distribution, meaning a path from start to finish was found quicker. This is surely because the guiding points are generated with greater probability closer to the target point. This leads the exploration graph closer to the target much quicker than if the points were randomly generated with equal probability anywhere in the world.

I chose 10 different small steps for experimentation ([4, 6, 10, 12, 15, 20, 25, 50, 75, 100]) and ran 10 trials for each one. Each trial was run with the same parameters (same world, starting point, target point, etc.), except I altered the seed for each one in order to generate different paths. As seen in the plots below, the number of RRT iterations decreases exponentially as the step size grows. On the other hand, there does not seem to be as clear of a correlation between the path length and step size. One could conclude that, in general, a larger step size leads to a shorter path, however at some point the step size does not affect the path length much, since there is only so much the length can decrease before we reach a close to optimal path. After running these trials, I would propose 15 as a good step size. With this value, the path length is close to optimal, the number of iterations is much lower than the step sizes of lesser values, and the world is explored more than those of greater value.



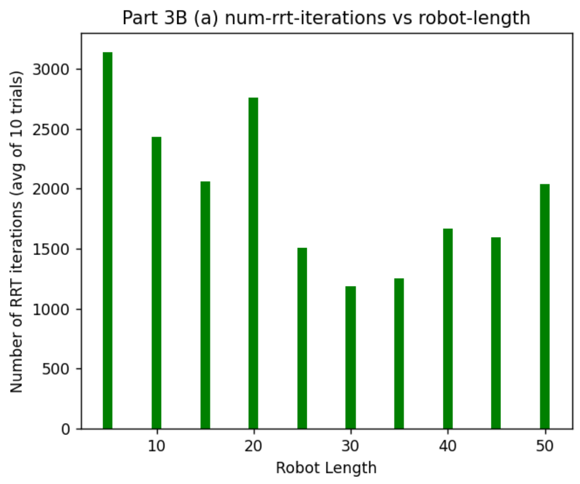


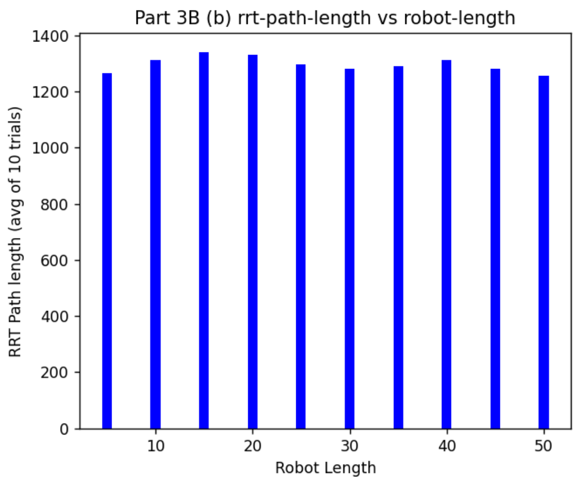
Line Robot

In this second part, I altered the RRT algorithm to run for line robots. To do this, I dilated the obstacles by the robot's length in order to avoid the robot colliding with obstacles, especially when turning. Same as the first part, guiding points are randomly generated using the python random library, with two options for distribution: uniform and Gaussian.

All line robots are located at a certain (x, y) point with orientation theta. At each RRT iteration, a guiding point p is chosen randomly (with specified distribution). As long as the robot will not hit an obstacle, it turns in the direction of the point p (this becomes the new theta) and moves by magnitude of step size. Below is a screenshot of one run, where the robot is lined in green, the exploration tree is in red, and the final path is outlined in black.

For the experiments in this part, I chose 10 different robot lengths ([5, 10, 15, 20, 25, 30, 35, 40, 45, 50]) and ran 10 different trials for each one. Similar to the trials that I ran in part 1, all parameters remained the same except for the robot length and the seed. Looking at the plots below, the optimal robot length is 30, since, at this value, the number of RRT iterations is minimized. It is interesting to note that the robot length does not seem to affect the path length in any meaningful way.


Part 3B (a) num-rrt-iterations vs robot-length
Number of RRT iterations (avg of 10 trials)
Robot Length


Part 3B (b) rrt-path-length vs robot-length
RRT Path length (avg of 10 trials)
Robot Length