Project 2

Will Graham

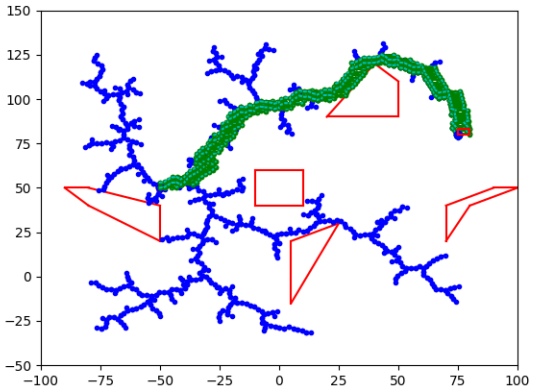
Problem 1

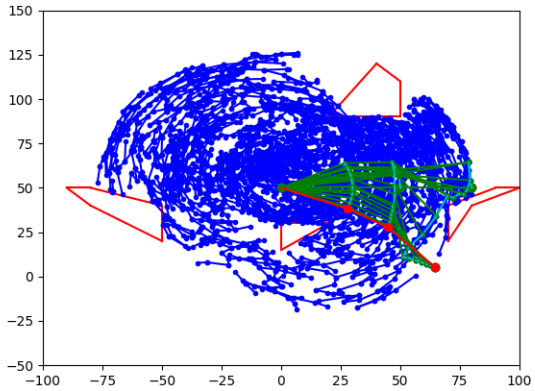
1.1 (20 pts) Implement an RRT. Use the two environments provided to plan a path from init to goal. Sample toward the goal instead of a random sample with 5% probability. Provide an image of both your tree and the resulting plan your algorithm finds for running on env0. Supply images of the resulting plan execution for env1.

For help debugging, our implementation produced a plan on env0 in about 1.8 seconds with a step size of 2. For env1 we used a step size of 0.15 and needed at least 5000 samples to get a plan, but sometimes more samples. This would sometimes take over 2 minutes to run.

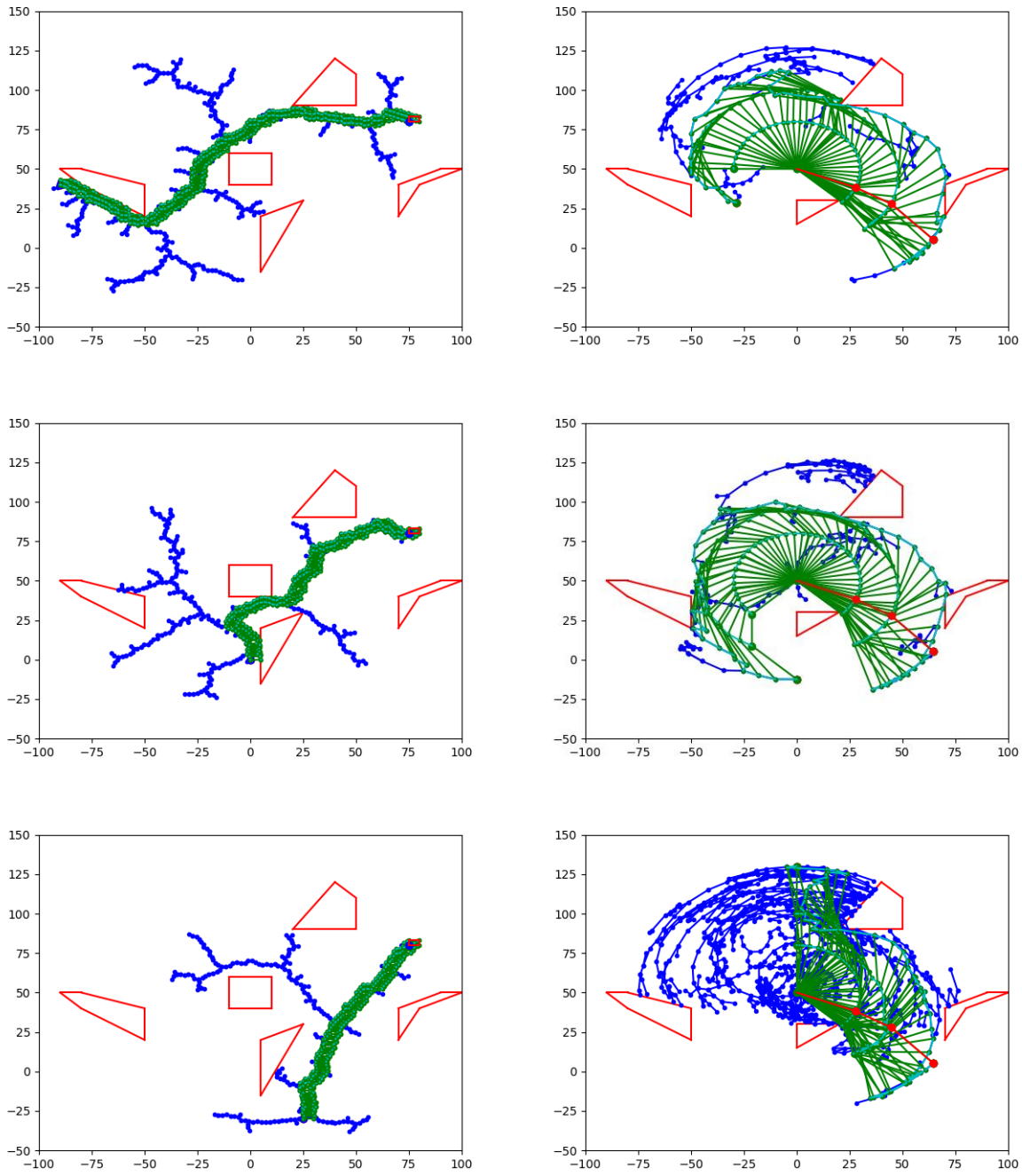
Run your RRT for both environments again with three, significantly different start and goal positions for each and again provide images of the tree and plan as before.

Env0 and Env 1 with 5% probability of sampling towards goal



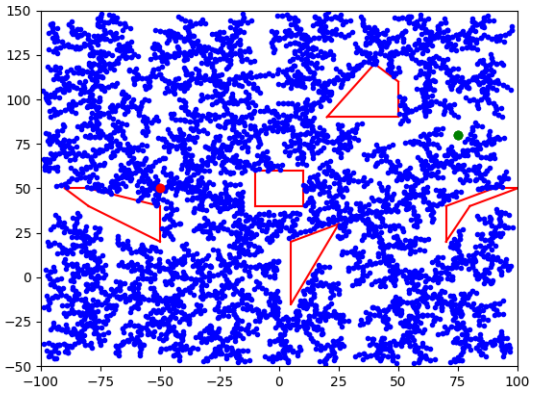


Additional Configurations (run with same probability and step size, included in zip folder)

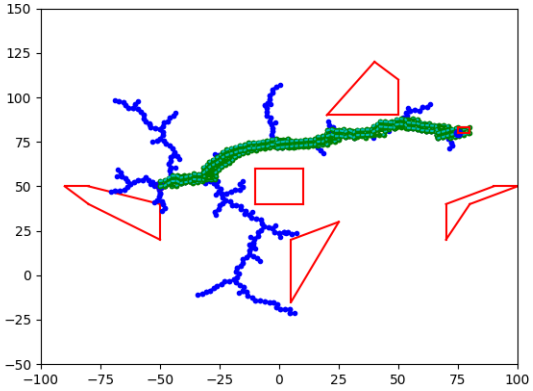


1.2 (10 pts) Compare the trees and plans produced by sampling toward the goal with 0%, 5%, 10%, and 50% probability for env0.

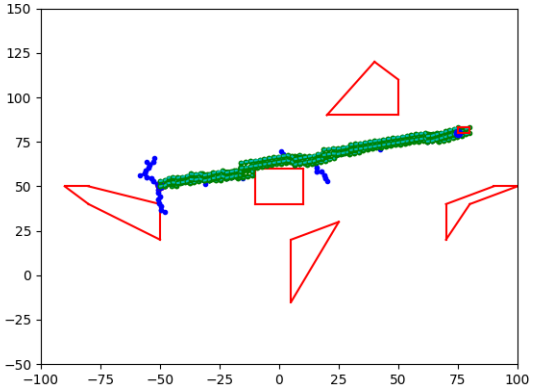
Env0 with 0% probability of sampling towards goal step size of 0.2



Env 0 with 10% probability of sampling towards goal step size of 0.2



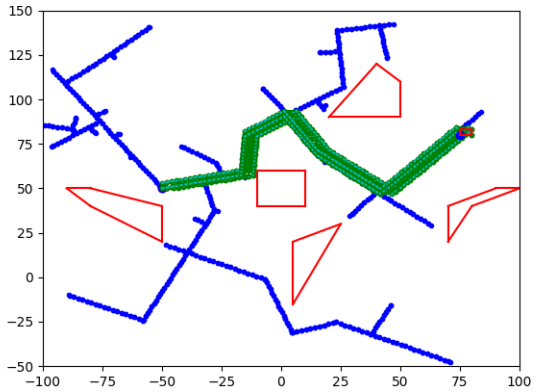
Env 0 with 50% probability of sampling towards goal step size of 0.2

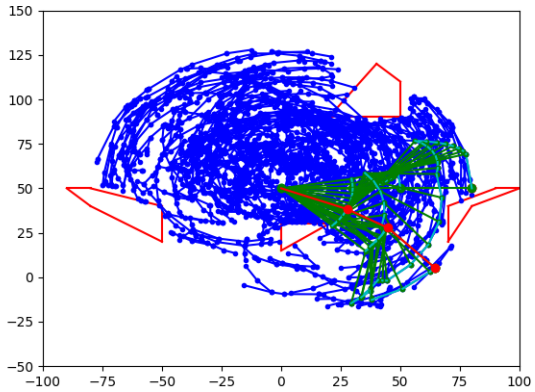


- The higher the percentage of sampling the goal, the more directed towards the goal. In our limited environment, this led to much quicker computing times. Unfortunately, this might also lead to our algorithm getting trapped in obstacles.

1.3 (10 pts) Extend your RRT implementation to create an RRT connect, replacing the standard `build_rrt()` function with `build_rrt_connect()` which continues to extend towards a given sample instead of only making a single ϵ length step until a collision occurs. Compare the resulting tree from running RRT-Connect to running the standard RRT for both env0 and env1. Use a 5% goal sampling bias for both versions. Again provide the images of resulting tree and plan found by your algorithm.

- Original environments

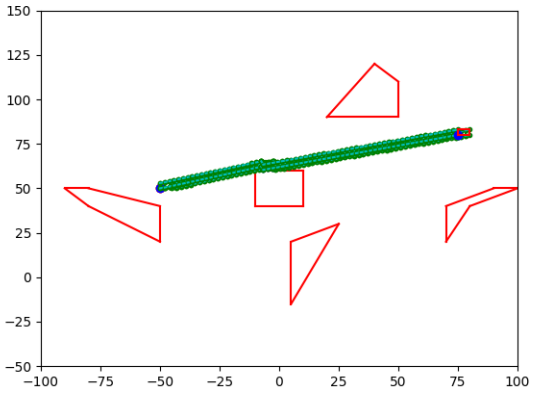


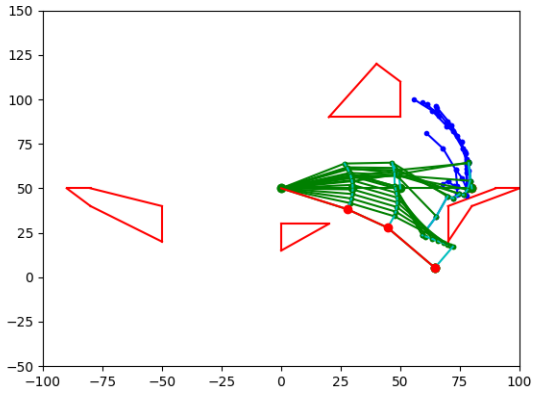


- Custom environments

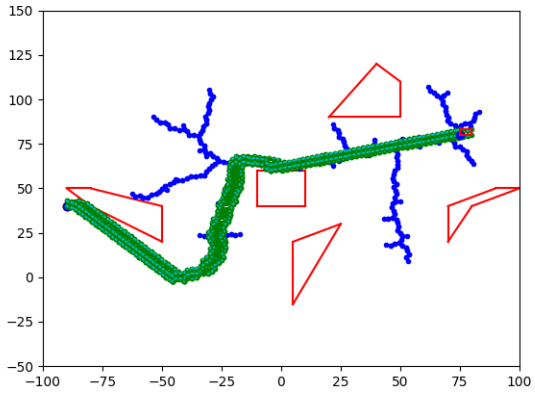
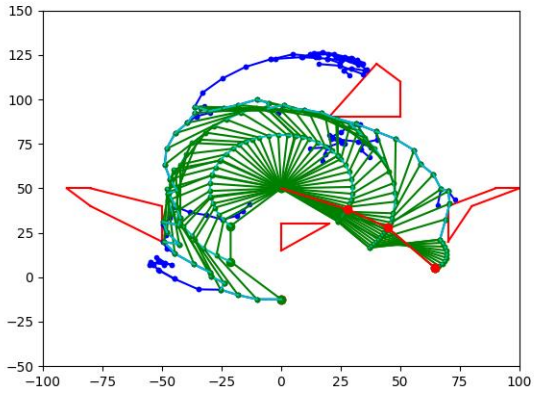
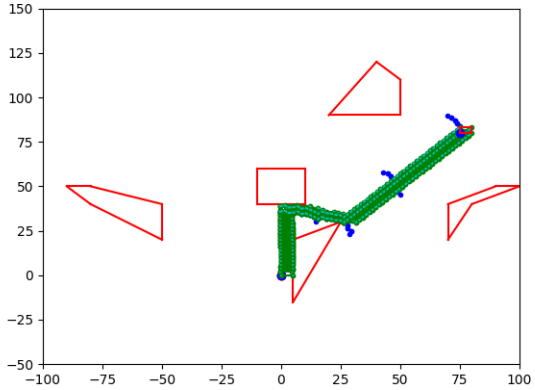
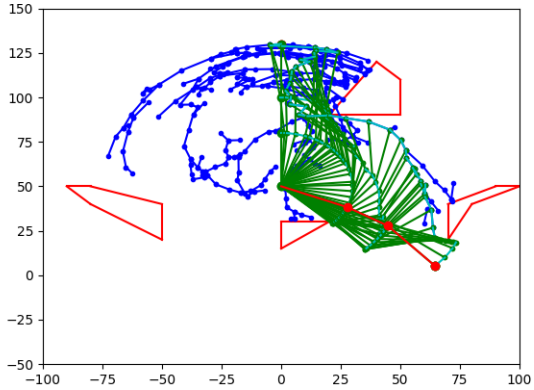
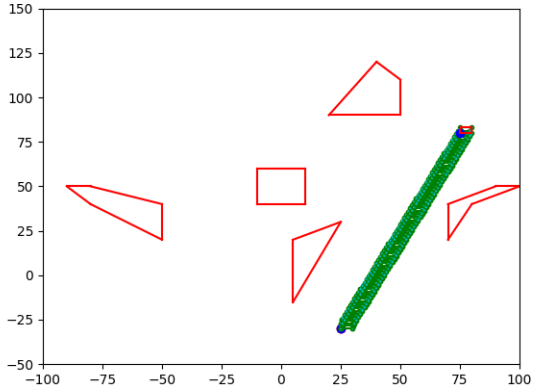
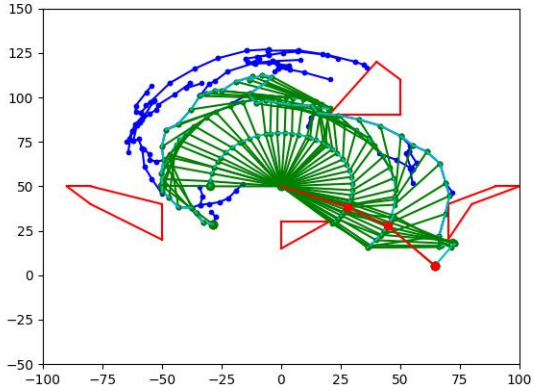
1.4 (10 pts) Now using your RRT connect implementation build a bi-directional RRT where you build two RRT-connects alternatively growing each one. Replace growing toward the goal with growing toward the other tree. Compare the resulting graph / trees and plan found to those found in the other versions above.

- Original Environments





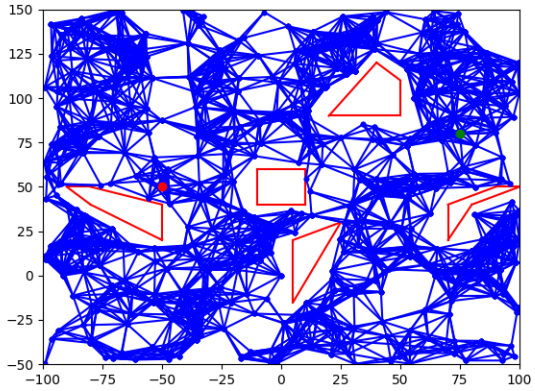
- Custom environments

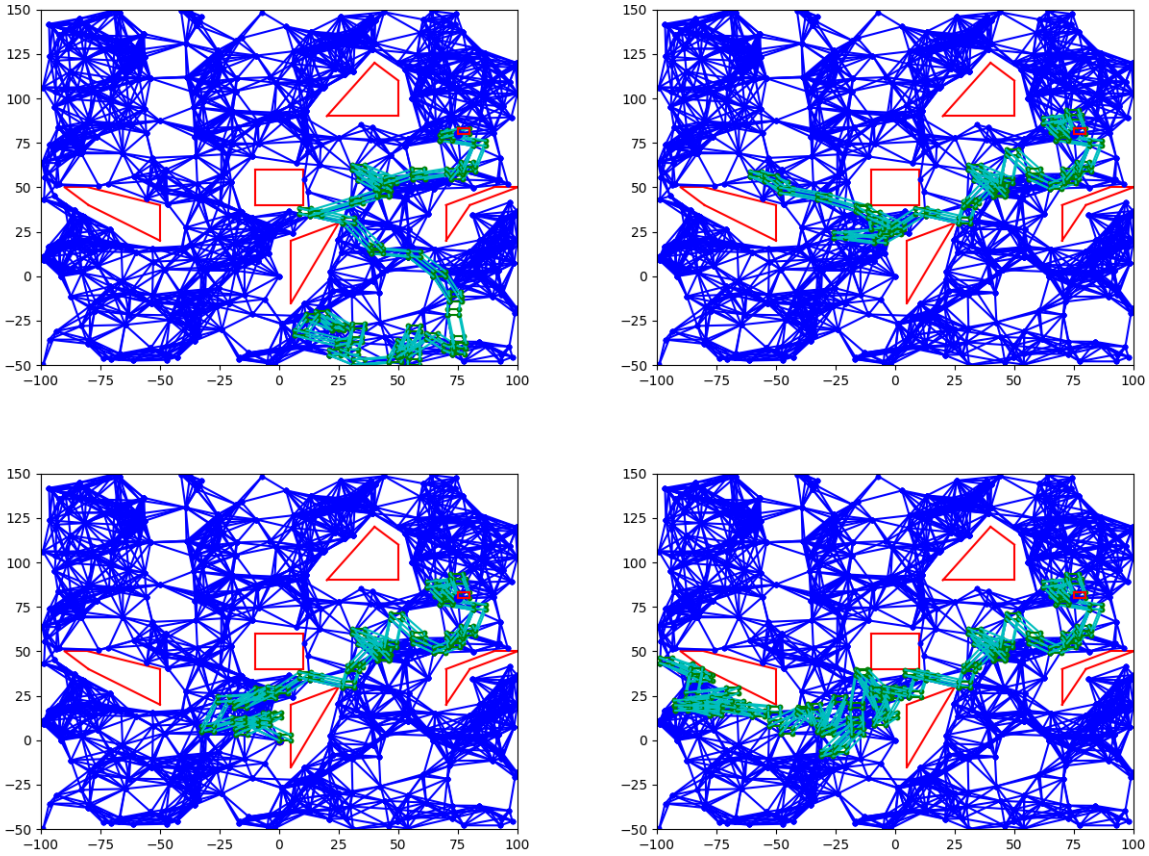
Problem 2

2.1 (30 pts) Implement the standard PRM using the same robots and environment as above. Build the PRM on the same maps, but now show results for planning with multiple start and goal locations, while building only a single roadmap. Show the roadmap built before attaching goals and initial positions, as well as the plans found for each of the different test situations. How does the resulting map change with different number of samples? How about changing the radius for Rdisk?

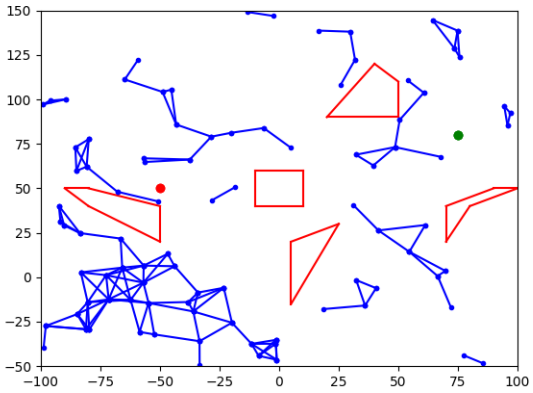
PRM on env0 with 500 samples, step size of 2 with radius of 25 and no plan shown

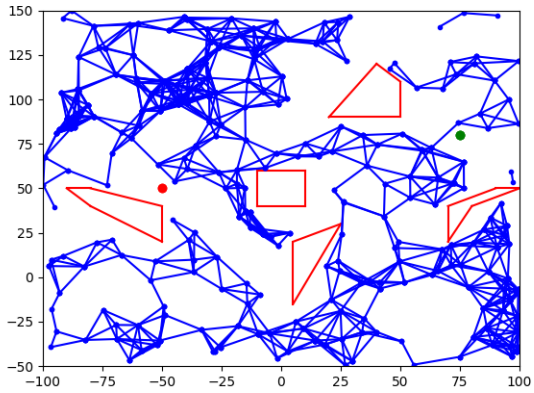


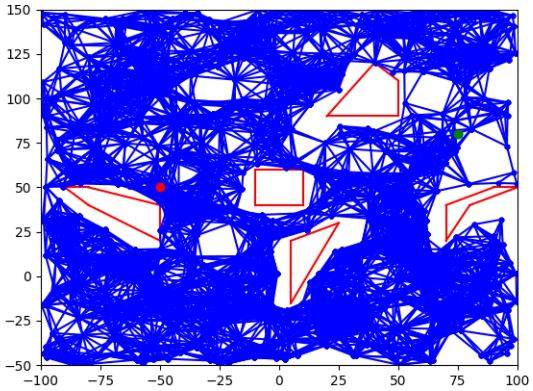
PRM on env0 with different starting points with same goal on same map



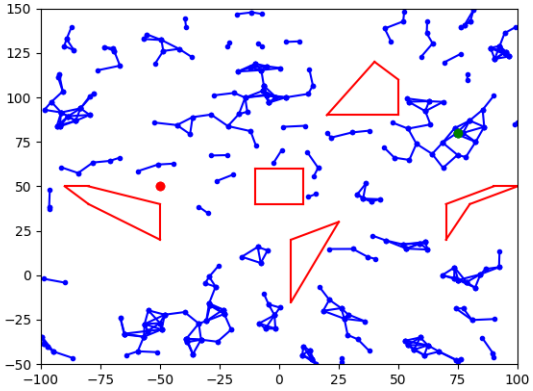
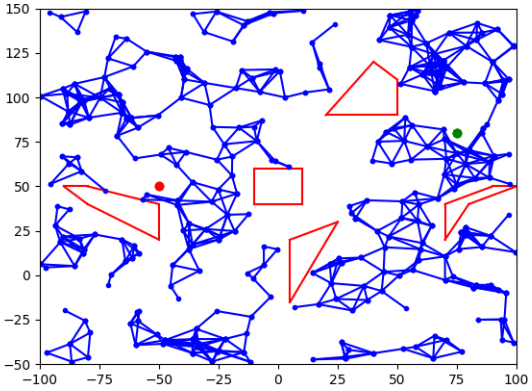
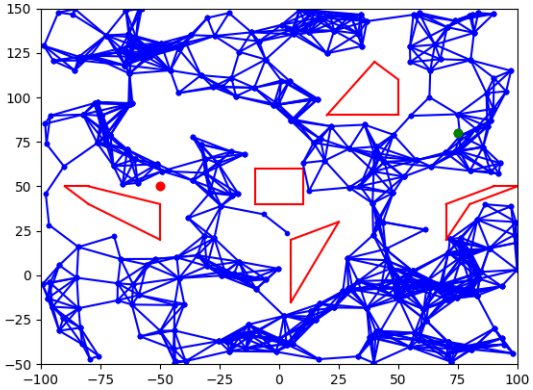
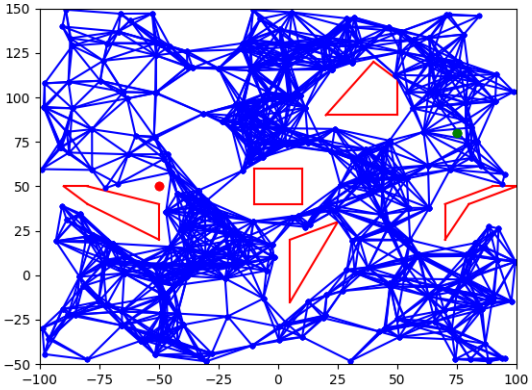
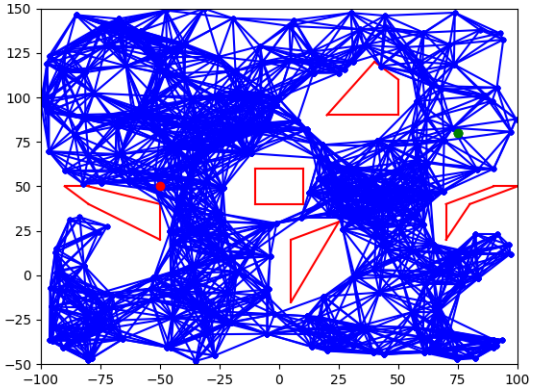
Different sample sizes (n = 100, n = 250, n = 500; all with step size 2, radius 25)



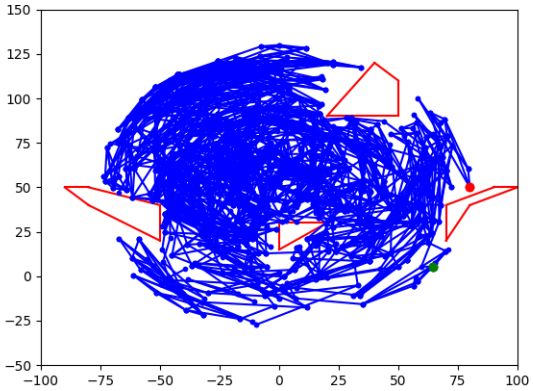
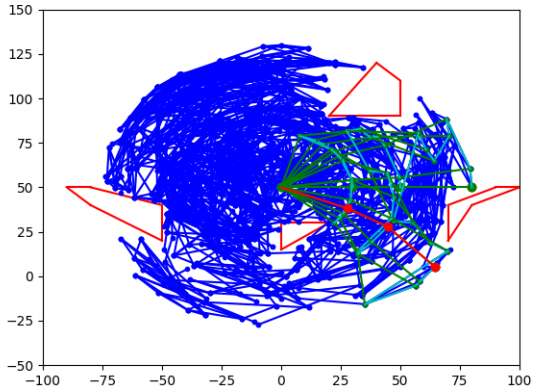
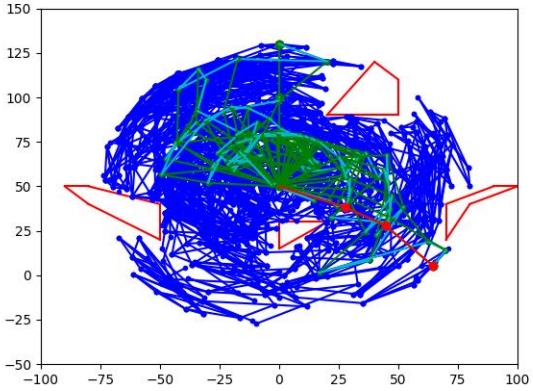
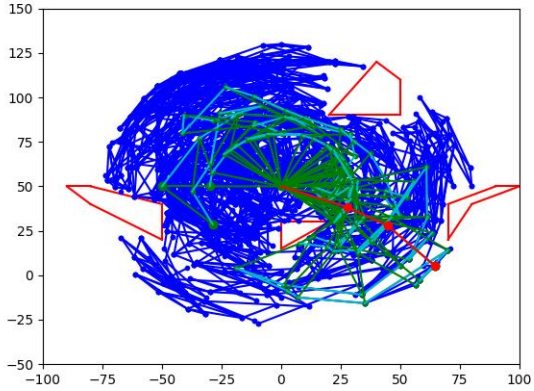
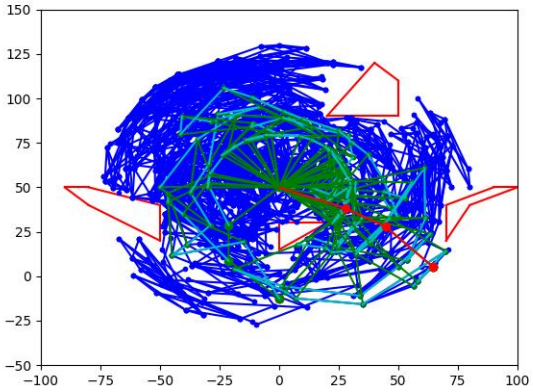




Different Radius Sizing (r = 10, 15, 20, 25, 30, with 300 samples)

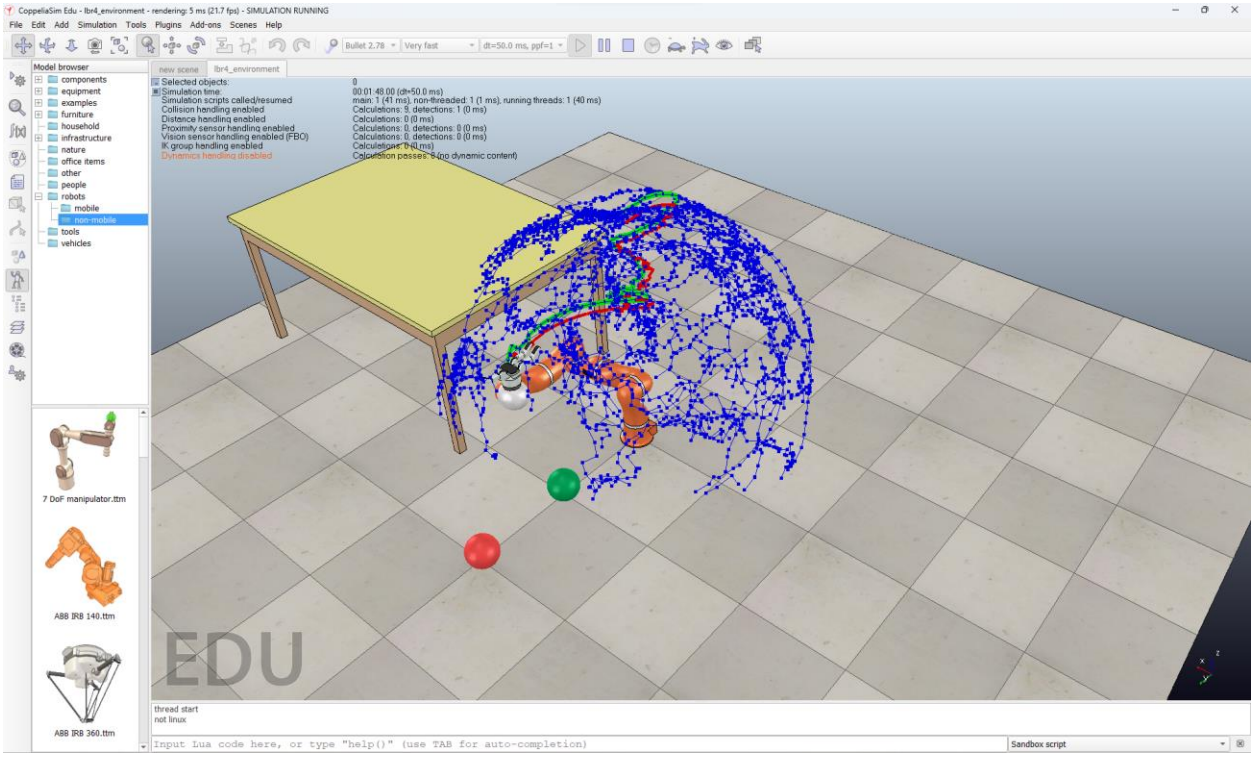






PRM on env1 with 750 samples, step size of 0.01, radius of .75

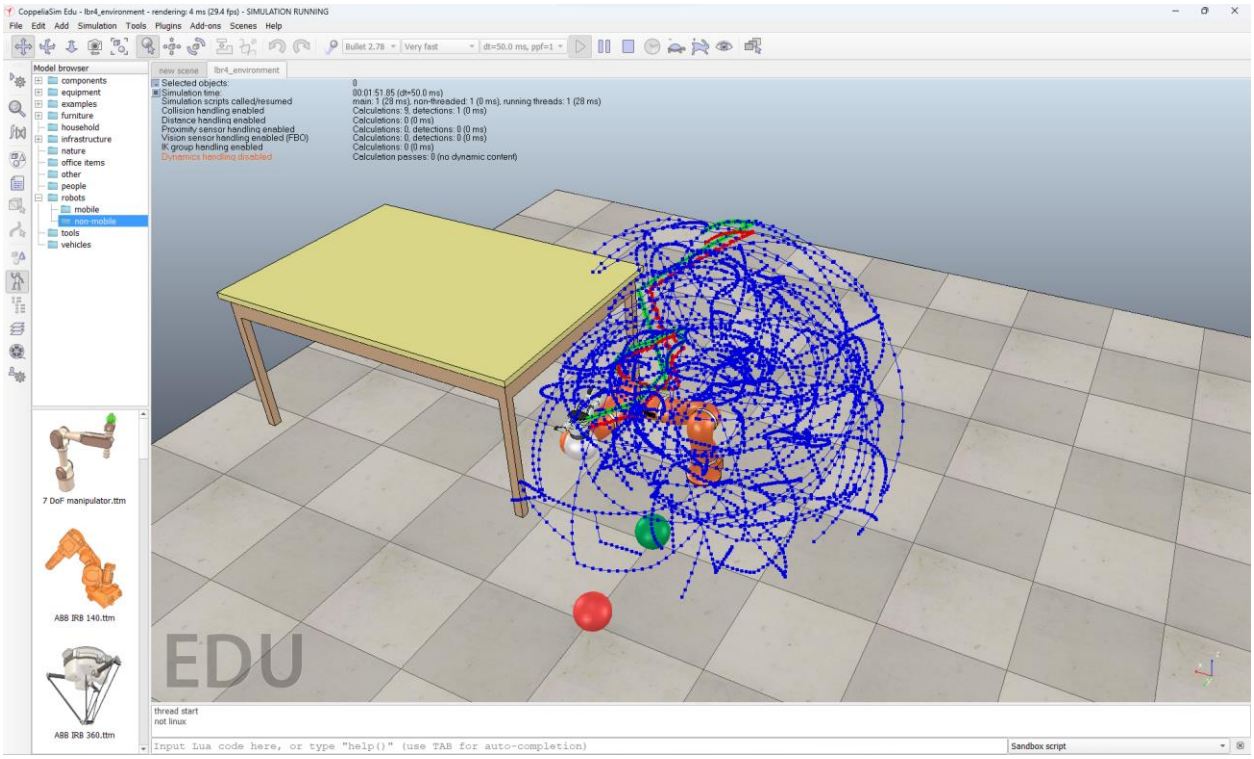






Problem 3

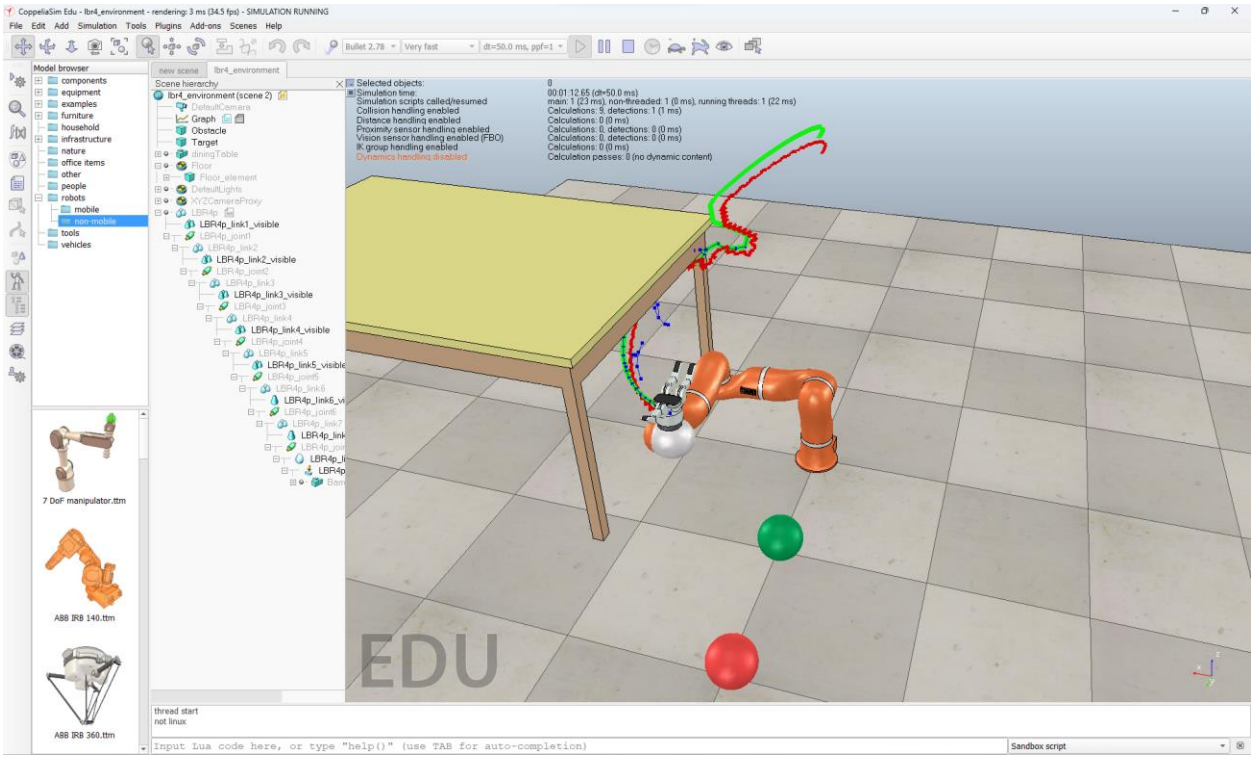
3.1 (5 pts) Run normal RRT with CoppeliaSim robot. Show screenshot of CoppeliaSim.



3.2 (5 pts) Run RRT connect with CoppeliaSim robot. Show screenshot of CoppeliaSim.



3.3 (5 pts) Run bidirectional RRT connect with CoppeliaSim robot. Show screenshot of CoppeliaSim.



Problem 4

4.1 (1 pt) What was the hardest part of the assignment for you?

4.2 (1 pt) What was the easiest part of the assignment for you?

4.3 (1 pt) What problem(s) helped further your understanding of the course material?

4.4 (1 pt) Did you feel any problems were tedious and not helpful to your understanding of the material?

4.5 (1 pt) What other feedback do you have about this homework?

1. The hardest part of this assignment was figuring out prm. I mistakenly put my second queue in a collision from the start, and I spent a good four hours looking for a bug that wasn't there. Additionally, I'm not positive that my RRT-connect worked exactly perfect with the Coppelia sim. It appeared to understand there were collisions, but the 3d simulation didn't appear to line up exactly with the collision function. I don't think that was necessarily my fault, but I'd definitely explore that more if we weren't only simulating these robots.
2. The RRT was easy for me. The collision and bidirectional were slightly more challenging, but the first RRT was relatively straightforward. I was also much better at interfacing with the given libraries/code.
3. I felt like the issue with a starting configuration ended up helping me understand the PRM algorithm much better. Since I had to go through it a bunch, I ended up spending far more time on it than I normally would have spent.
4. There were some questions with the Coppelia Sim, but I don't think that necessarily made my this any more difficult. I think those aren't related to the material within the class. Overall I felt like I learned a ton from this assignment.
5. Great assignment, and I liked it a lot! I was genuinely looking forward to working on the problems and I'm excited to turn it in.