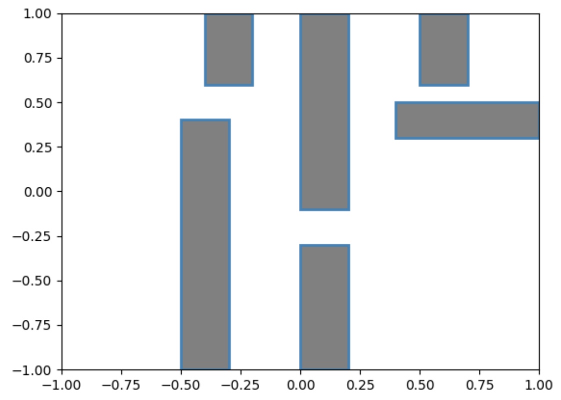
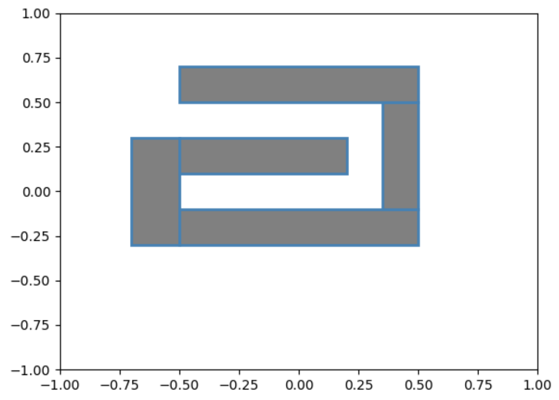
Part 1 For this assignment, an RTP planner was implemented. The RTP planner is a randomized tree planner that selects a random configuration, q_a from the existing tree and a random configuration, q_b from the configuration space, that has a small probability of being the goal. If a straight line between q_a and q_b can be drawn, it is added to the existing tree. This planner was tested with two user-defined robots and environments. Lastly, the implemented RTP planner was benchmarked against the existing planners in a series of prescribed environments.

Part 2 Two robots were created: a point robot and a square robot. The point robot is a zero-dimensional robot that can translate in the x-direction and y-direction. The point robot had a configuration space of $\mathbb{R}^2$. The square robot is two-dimensional with a side length of 0.05 and can translate in the x-direction and y-direction, as well as rotate in a two-dimensional plane. The square robot had a configuration space of $SE(2)$.

Part 3 In figures 1a and 1b below, environments 1 and 2 are shown.



(a) Environment 1



(b) Environment 2

In environment 1, an environment with thin passages between various regions was constructed. For environment 1, a start point was placed in the leftmost passage, while the goal was placed in the upper-right corner. This choice of start and goal forces the planner to navigate through the narrow passage ways to reach the goal.

In environment 2, a bug trap environment was constructed. The start point was placed inside the trap, while the goal point was placed outside the trap. This environment choice was created to force the planner to navigate around obstacles that in the direction away from the goal.

Part 4 Below in Figure 2, the path generated by the RTP planner for the point robot in environment 1 is shown.

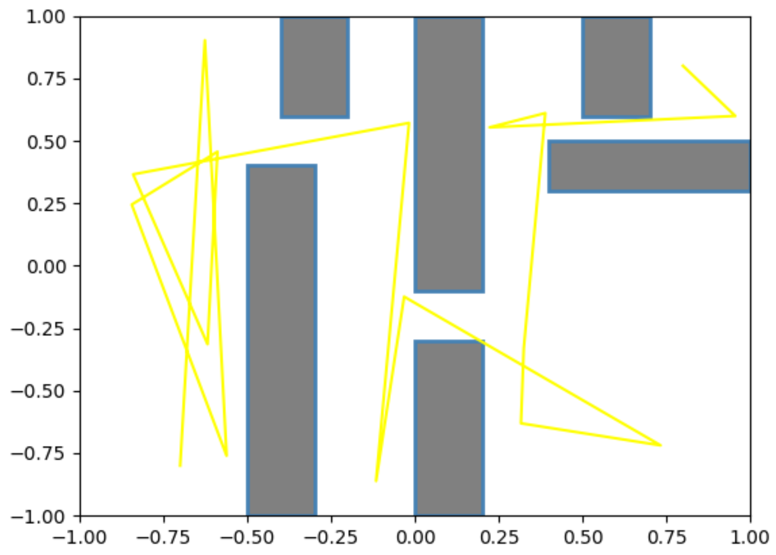


Figure 2. Plath generated for the point robot in environment 1

Next, in figure 3, the path generated by the RTP planner for the square robot in environment 2 is shown.

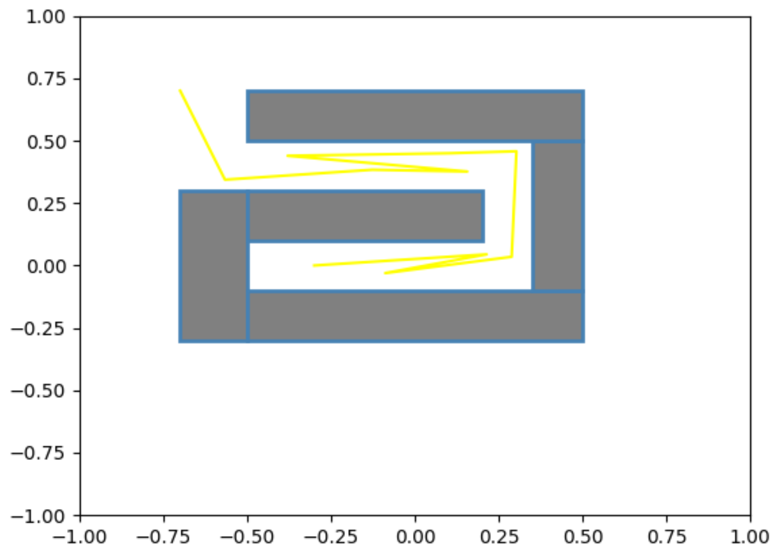
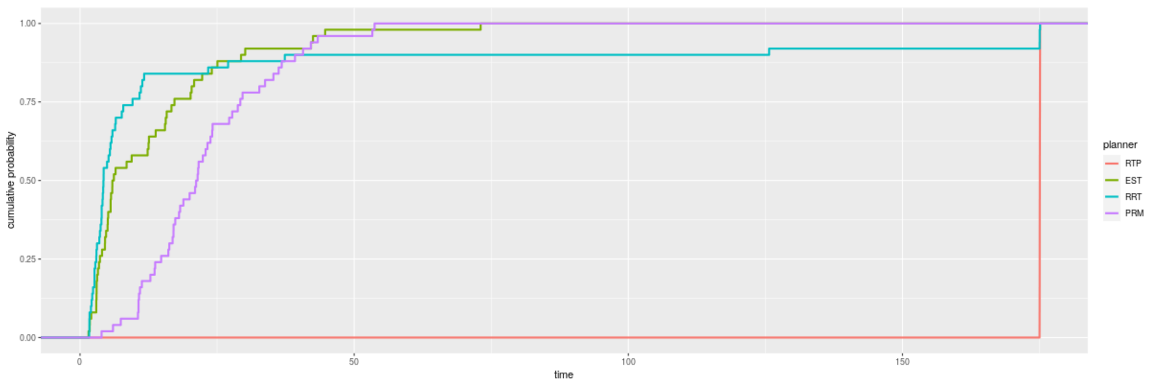
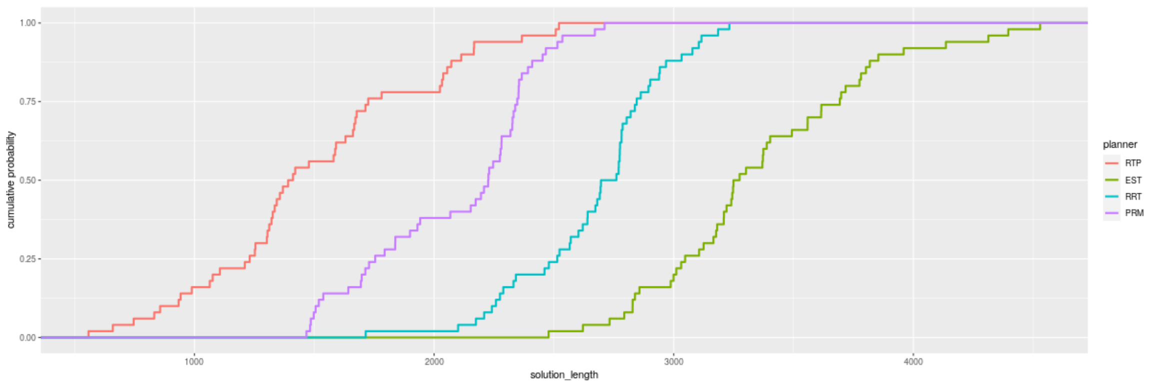


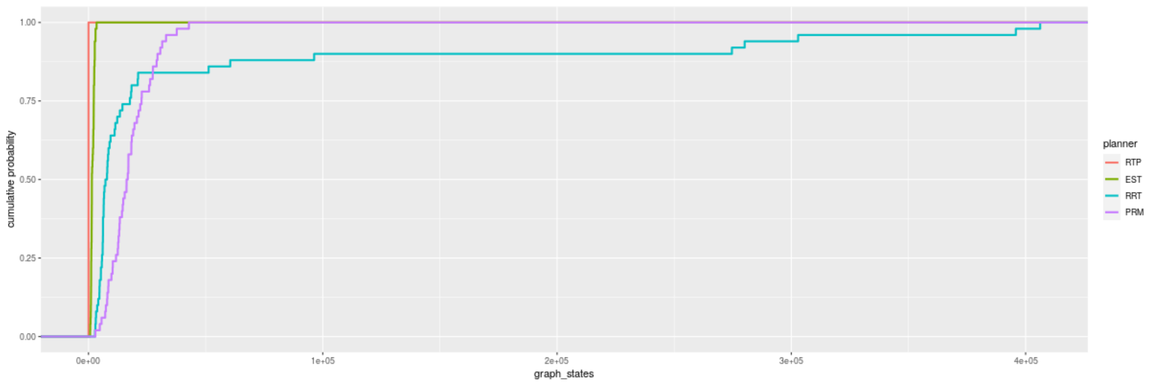
Figure 3. Plath generated for the square robot in environment 2

Part 5 The RTP planner overall performed worse than the other planners. This was expected, as RTP was designed to be a bare-bones "basic" planner. The other planners, meanwhile, are more advanced, such as RRT serving as an upgrade to RTP with its intermediate states and nearest-neighbors data structure.

The following figures show the benchmarking results for the four planners of interest in the Home environment:

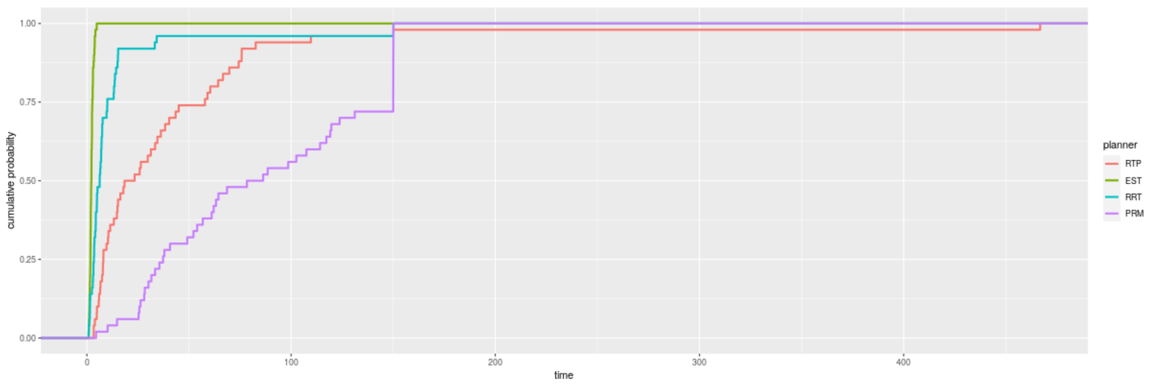


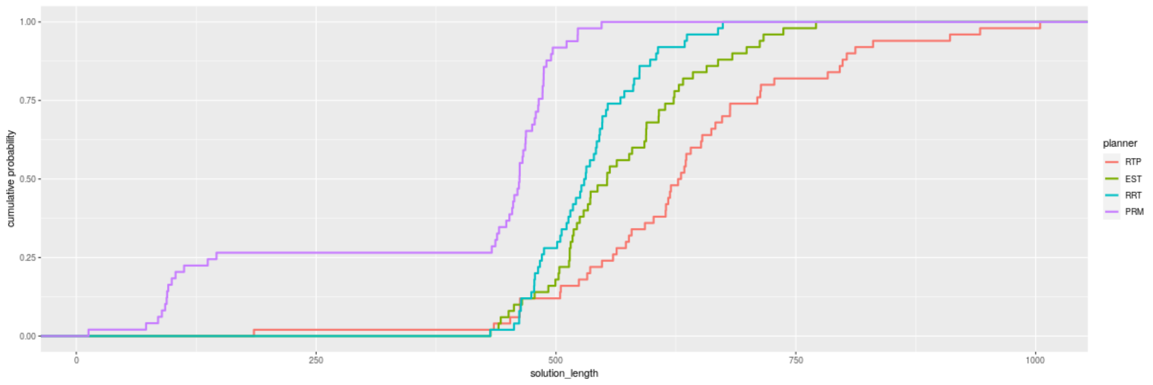


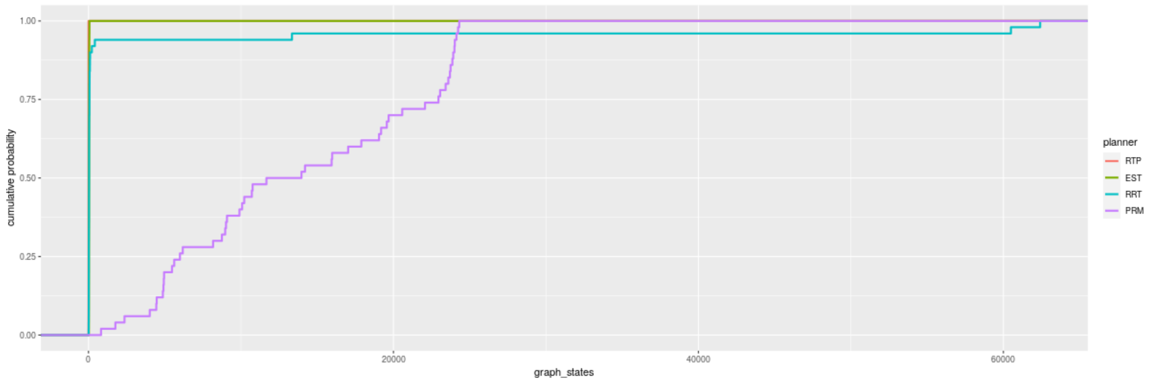


As can be seen, the RTP planner took a very, very long time to solve in the environment. This is due to the randomness of the paths found as it finds a solution. However, surprisingly enough, it overall had shorter solution lengths when compared to the other planners. This implies that in this particular environment, the randomness of the RTP planner aided in finding shorter paths in the Home environment as opposed to finding a correct answer using intermediate states or "nearest-neighbors" like RRT. It also had a low number of graph states needed before finding a solution. While this might indicate some issues in the code, it may also indicate that it required a lower number of samples than the other planners. This makes sense as the other planners like RRT use intermediate states.

Unfortunately, time constraints prevented us from performing a proper benchmark for the Apartment environment. Here were our initial results:







While these results don't match our desired output, we can still see some of the drawbacks from the RTP planner. For example, the RTP planner had much longer solution paths than the other planners. However, those benchmarks also show that RTP performed faster than RRT at points, which isn't what was supposed to happen. We would expect the time graph for the Apartment environment would follow a similar pattern to that for Home where RTP guarantees a solution after an extremely long time. We were able to fix these problems in the code, but we do not have the benchmark results for this environment to show for it.

Part 6 Since we both contributed to the same parts of the assignments (as opposed to Project 2, where we simply assigned problems to ourselves), here are our individual rantings:

Santi:

(a) Implementing RTP: This one took a while due to the lack of familiarity with the library and getting re-accustomed to C++ coding. That being said, the instructor hint of using RRT as a template helped out a lot, since that allowed me to look through the code and comment as much as I could so I could get an understanding and make something that mostly worked. 7/10, spent $\sim$ 4 hours on this.

(b) Exercise 2: Personally, this was the hardest exercise due to the need to visualize the space, even if I did not perform this. However, it was still quite difficult due to planBox needing a different implementation than planPoint, and because we weren't using a "template" like for the RTP implementations, then I was mostly working blind. 8/10, spent ∼ 4 hours on this.

(c) Exercise 3: By the time I started working on Exercise 3, I had an idea of what was being asked for, and I also followed the benchmarking file from the OMPL library closely to know how to format the code. Still took a while to perform the initial round of debugging, but it was easy to work out. 6/10, spent ∼ 1 hour on this.

Eli:

(a) Implementing RTP: This took some time to debug and remove the proper functionality and ensure correctness. 6.5/10 difficulty, and spend about ∼ 2 hours understanding and debugging existing code.

(b) Exercise 2: Took some time to visualize the space and ensure the planners were set up properly. 7/10 difficulty, with about ∼ 3.5 hours of work.

(c) Exercise 3: Not terribly difficult once the script was written, but took an unreasonable amount of time to get meaningful results, since my local computing resources are underpowered for this application. 9.5/10 difficulty and ∼ 3 hours of total time.

And overall, our individual contributions were as follows:

Santi –

- Wrote the first draft of RTP.cpp and RTP.h by taking RRT.cpp and RRT.h from the OMPL library, commenting each line for general understanding, and removing any code that isn't needed.
- Implemented planBox and makeEnvironment2 in Exercise2.
- Wrote the entirety of Exercise 3 and debugged it so it would compile. Performed the first set of benchmarks.
- Performed minor debugging for Exercise 2.
- Made code comments for all files.

Eli –

- Created the visualizations of the environments and planner solution paths.
- Performed general debugging of the code in Exercise 2 and the RTP implementations. Ensured the RTP implementations and Exercise 2 were all compiled successfully.
- Performed general debugging of Exercise 3 and adjusted benchmark parameters to obtain meaningful solutions.