Miles Sigel, Alex Prucka

- (2.5 points) A succinct statement of the problem that you solved.

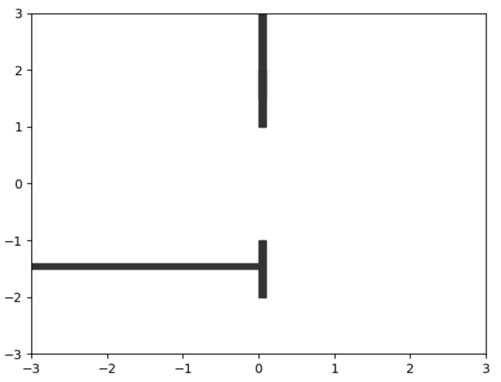
For project 3, our group had to implement the RTP planner using the OMPL class structure and hierarchy used to implement other planners. In order to do this, we had to identify the unique aspects of RTP with respect to other planners such as RRT, in order to make a working planner and gain experience working with the underlying OMPL code instead of just interfaces. We additionally had to solve the problem of creating environments with low-level primitives and plotting libraries. Finally, we had to benchmark our solutions in order to make generalizations about its effectiveness.

- (2.5 points) A short description of the robots (their geometry) and configuration spaces you tested in exercise 2.

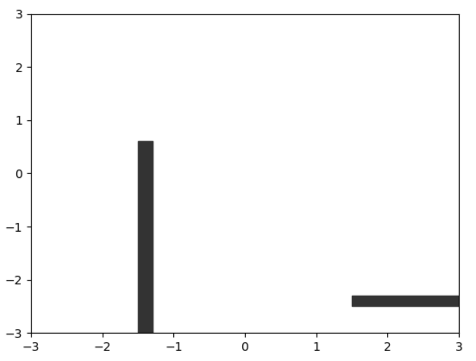
We had two robots, a point robot, and a box robot. Our point robot that could only translate in the plane had a c-space of R2 and was bounded by a (3, 3) box. The other robot was a box robot in SE2, meaning that it could translate and rotate within the plane. The box robot had a side length of 0.5.

- (5 points) Images of your environments and a description of the start-goal queries you tested in exercise 2.

ENV 2

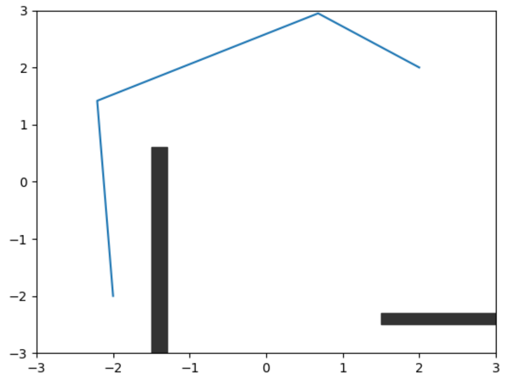


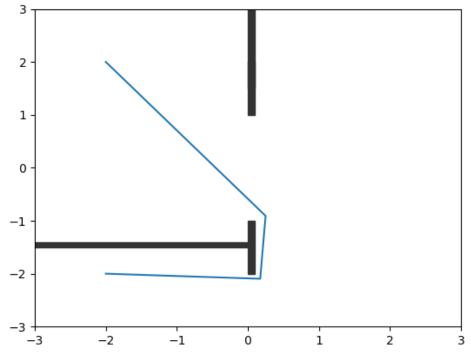
ENV 1

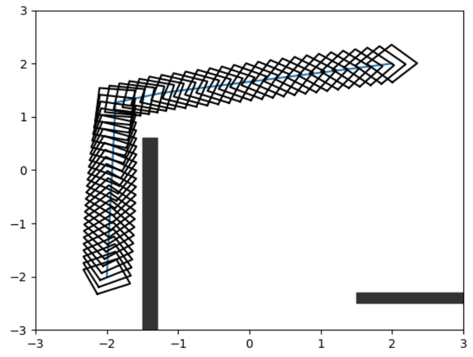


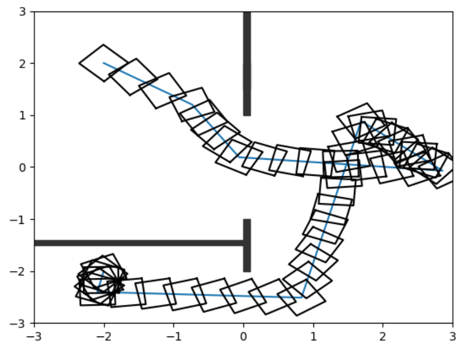
Both the start and end queries are different for the two environments. For ENV1, the start is (-2, -2) and the end is (2, 2). For ENV2, the start is (-2, -2) and the end is (-2, 2).

- (5 points) Images of paths generated by RTP in your environments you tested in exercise 2.







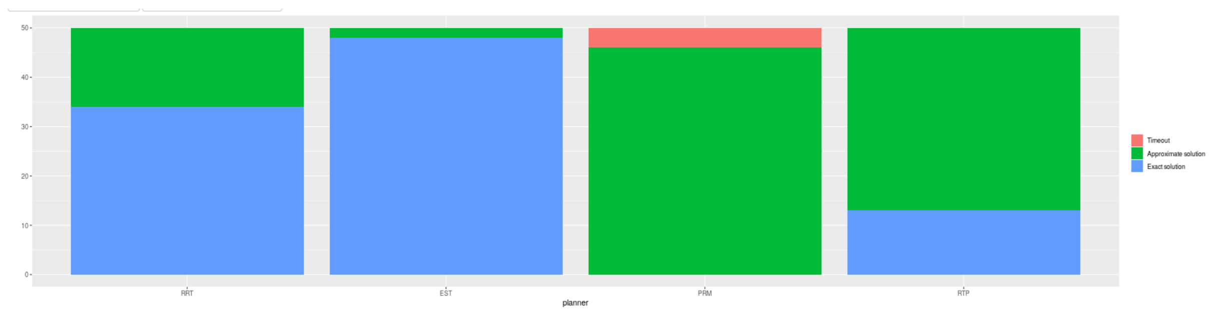


- (5 points) Summarize your experience in implementing the planner and testing in exercise 2. How would you characterize the performance of your planner in these instances? What do the solution paths look like?

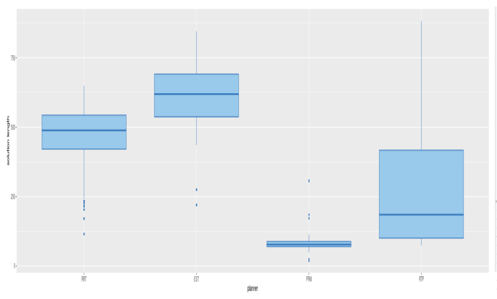
I believe that our planner did exceptionally well given the circumstances. Given that our environments were not super complex, there were non-trivial parts to them, especially in environment two in navigating through the narrow passage. One part of the random-based planner that is non-optimal and apparent is in the box environment 2 path where it seems to bounce back in forth in open space a little for no apparent reason. Overall, the experience was super helpful working with coding skills in python and c++ to both graphs and create the planning sequence.

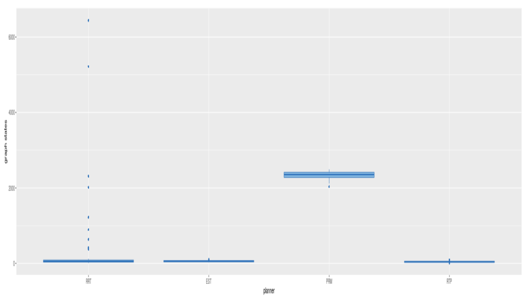
- (15 points) Compare and contrast the solutions of your RTP with the PRM, EST, and RRT planners from exercise 3. Elaborate on the performance of your RTP. Conclusions *must* be presented quantitatively from the benchmark data. Consider the following metrics: computation time, path length, and the number of states sampled (graph states).

  On the apartment environment, our implementation of RTP worked better than PRM, but worse than RRT and EST as shown in the status graph below. For all of the graphs below, the order of planners is RRT, EST, PRM, RTP (as the labels are tiny)

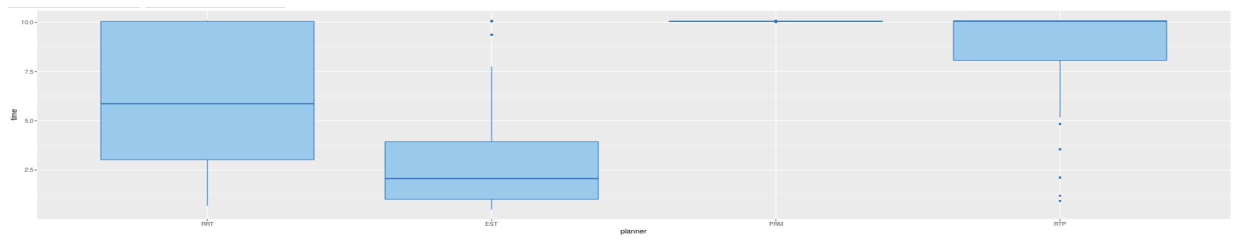


  The solution length for RTP was slightly better than average, however since it only found the solution 30% of the time, many of the approximate solutions may be shortening the average solution length and thus not truly being representative of shorter solutions. This is shown below on the left. It far outperformed PRM in terms of graph states, although EST also had a similarly low number of graph states. RRT tended to have few graph states but did have outliers with extremely large state counts. RTP ties for first in this category. Graph states are shown below on the right.

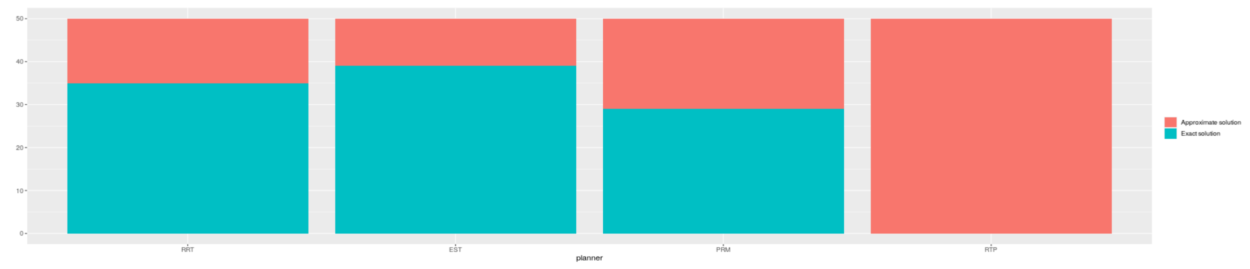




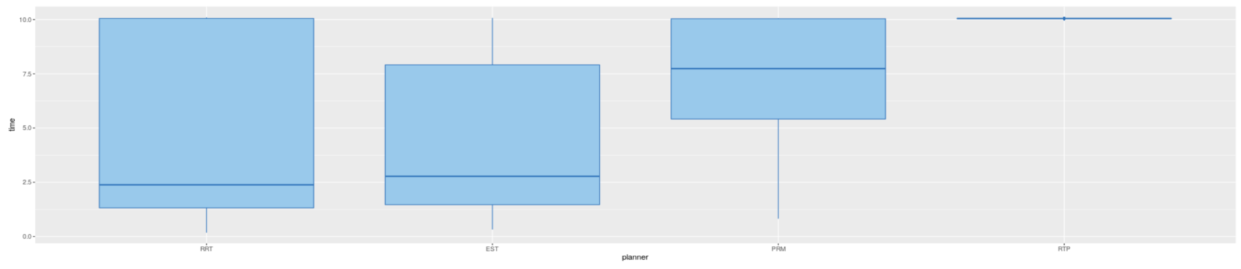
  In terms of solution time, RTP falls behind all but PRM. Occasionally it was very quick in finding a solution, but often it would take all the allotted time before giving an approximate solution and timing out.



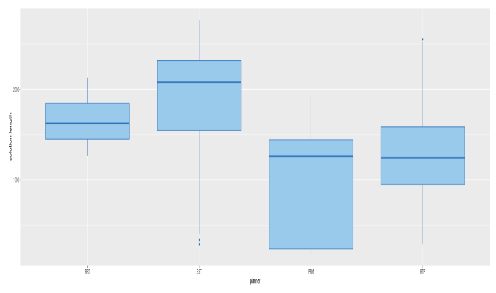
Performance on the Home environment was not nearly so promising. It managed to only find an approximate solution, while the other planners found solutions >50% of the time.

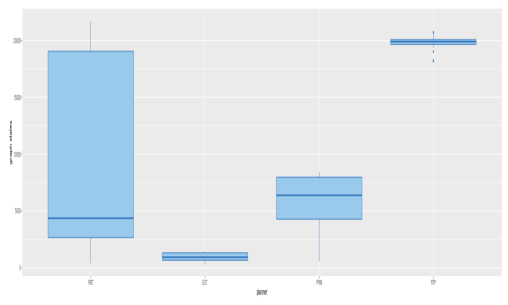


The time to solution graph looks as expected, with RTP timing out each and every time, whereas RRT and EST take an average of 2.5 mins, and PRM takes an average of 7.5 mins.



The solution lengths for all 4 planners were comparable in the home environment, with only PRM looking to have an edge in solution length. This is shown below on the left. Another terrible metric in the home environment for RTP is graph states. It constantly averaged the highest state count, nearly 20 times the states of EST. EST had by far the lowest amount of states. This is shown below on the right





RTP did manage to perform better than PRM on the apartment environment, but that was its only victory. It was worse than all three of the other planners in the Home environment, and it is very clear that this is a simple and not particularly smart path planning algorithm.

- (5 points) Rate the difficulty of each exercise on a scale of 1–10 (1 being trivial, 10 being impossible). Give an estimate of how many hours you spent on each exercise, and detail what was the hardest part of the assignment. Additionally, describe your individual contribution to the project.

1. Exercise 1 - I would give this a 8/10 in difficulty. While the reference and direction towards using RRT to help us implement this planner was incredibly helpful, there were many hurtiles that we had to jump over to understand c++. Having a good understanding of the overall algorithm, however, allowed us to quickly find the parts of the RRT planner that we would need to remove and what parts we would need to insert for the RTP. Using a vector backing to hold all the points was another source of understanding and something that we finally realized after many different configurations and attempts. Overall, this exercise was done by both Alex Prucka and Miles Sigel, and took around 6-8 hours to both understand and complete the exercise.
2. The second exercise involved creating and graphing environments for our robots. Creating the actual environments was not too difficult (3/10) since we have had experience with creating obstacles in the past. This took Alex Prucka only took around an hour to complete. The graphing was a little more complicated since we had to develop our own small wrapper around the matplot lib library, one that we had experience with. After configuring all the code and creating a process script to work through the .txt file, the overall python file is relatively straightforward. This took around 2 hours to complete by miles and was around (5.5/10) difficulty to complete.
3. Exercise 3 was not too difficult, and was mainly made up of referencing other benchmarking examples, and using the syntax and code forms found within to test our RTP example. I would rate it a difficulty of (4/10), as the benchmarking and comparisons is something we have practiced in the past projects. The component took about 1 hour to write the benchmarking code and about 1 hour to analyze the results, with 20 minutes taken to run the benchmarks themselves.

Contribution:

Exercise 1 - Miles and Alex

Exercise 2 - Miles and Alex

Exercise 3 - Alex