# COMP 550 Project 3 Report

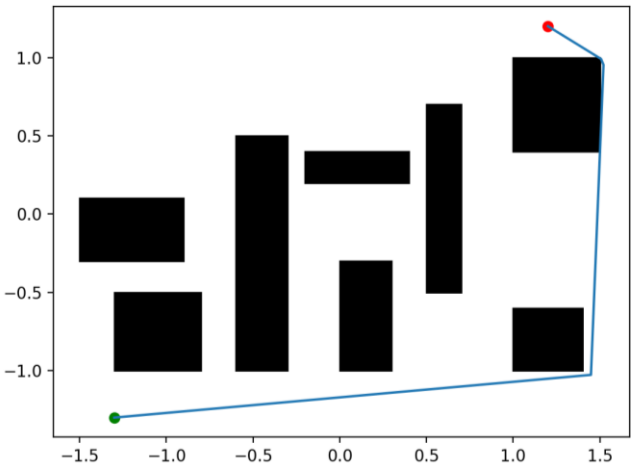
Hao Ding (hd25) & Haoran Liang (hl74)

- A succinct statement of the problem that you solved.
  - In Exercise 1, we implemented the Random Tree Planner algorithm mostly based on the framework of the RRT algorithm.
  - In Exercise 2, we designed two different work spaces to test our RTP algorithm, in each robot, we tested both point robot and square robot, and then visualized the path solution using Python.
  - In Exercise 3, we benchmarked our implementation of RTP in Cubicles and Twistycool scenarios with other three planners PRM, EST, and RRT. We generated plot images based on the benchmark results data file and analysis the performance of our RTP planner.

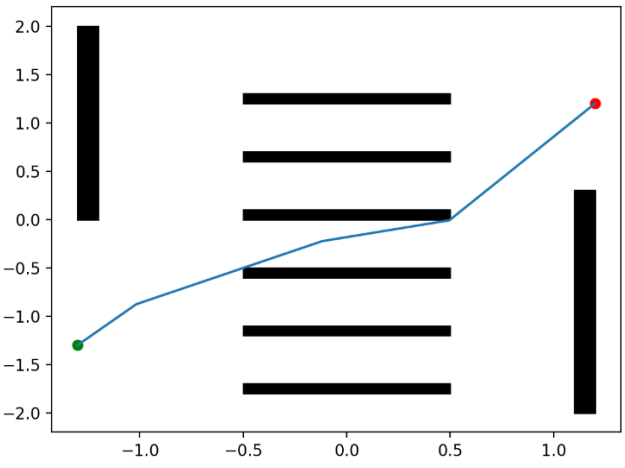
- A short description of the robots (their geometry) and configuration spaces you tested in exercise 2.
  - Point robot: The point robot has 2 parameters: x and y. Based on x and y, we can accurately locate the point in the work space. Therefore, it has 2 degrees of freedom and its configuration space is **R^2** and looks the same with the work space.
  - Square robot: The square robot has 3 parameters: (x, y) of the center and its rotation angle. We can use x and y to locate the square robot in the work space and use angel to figure out its direction. Therefore, it has 3 degrees of freedom and its configuration space is **SE^2**.

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

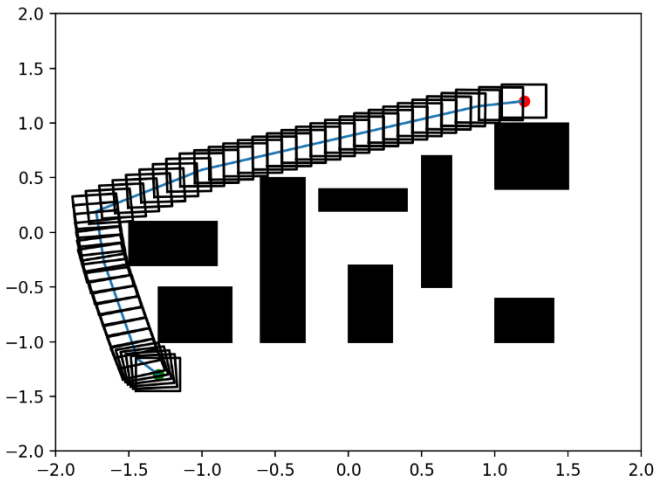
  Images of paths generated by RTP in your environments you tested in exercise 2.



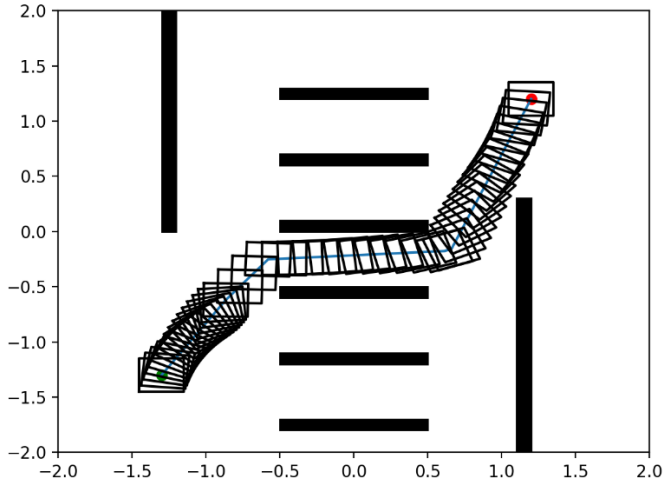

Point robot in Environment 1




Point robot in Environment 2




Square robot in Environment 1



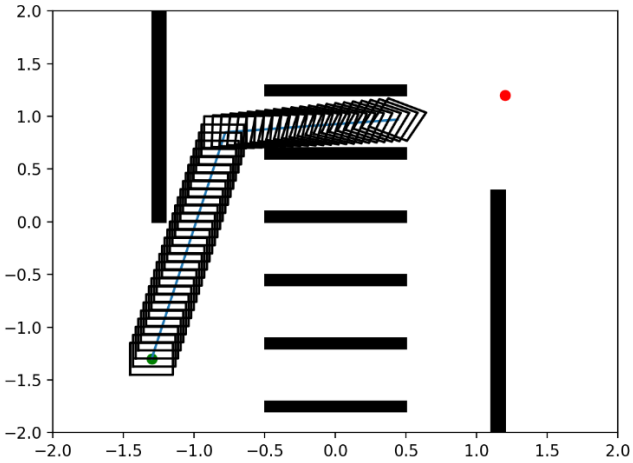

Square robot in Environment 2

The **green** point is the start point in images above, whose coordinate is **(-1.3, -1.3)**.

The **red** point is the goal point in images above, whose coordinate is **(1.2, 1.2)**.

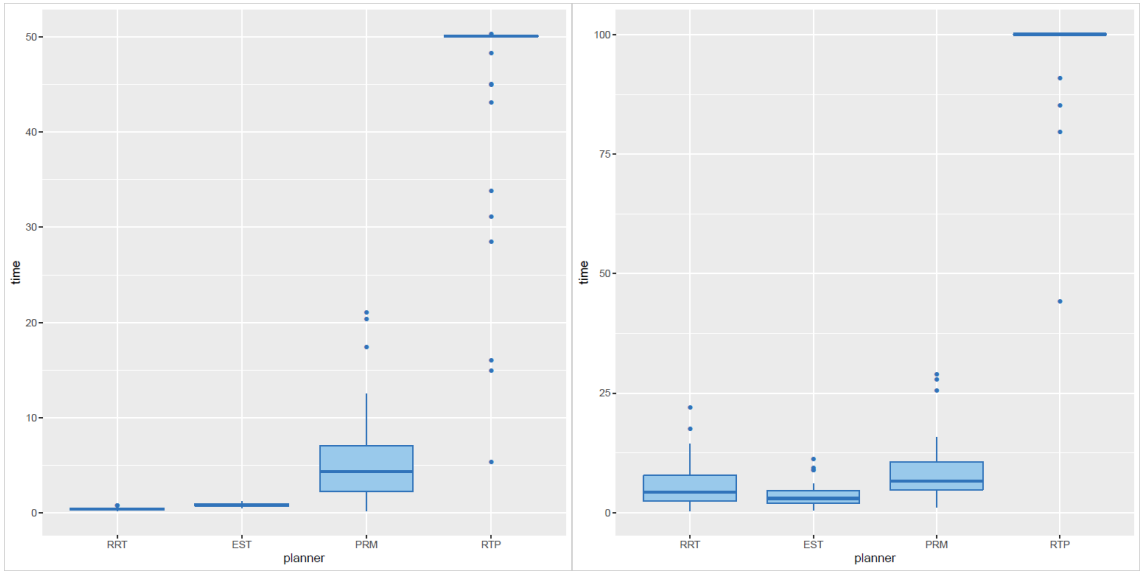
- Summarize your experience in implementing the planner and testing in exercise 2. How would you characterize the performance of your planner in these instances? What do the solution paths look like?
  - Most of our work is based on the existing RRT algorithm framework and the hand on solution file in the class, our RTP planner is almost the simplified RRT planner and 60% of our testing code is from the hand on solution code. Therefore, we spent a lot of time reading the code and understanding the logic behind it, to remain/update the core content and delete unnecessary parts.
  - The most distinctive feature is that the RTP planner's solution is **pretty random** and **can't** guarantee that it can reach the goal **in given time**. This is because that the growing process of the state tree is a random process, each time we randomly choose an existing tree node to extend the tree.

    For example, in the image below we tested the square robot in environment 2, our planner can't reach the goal in given time and stop in the middle.



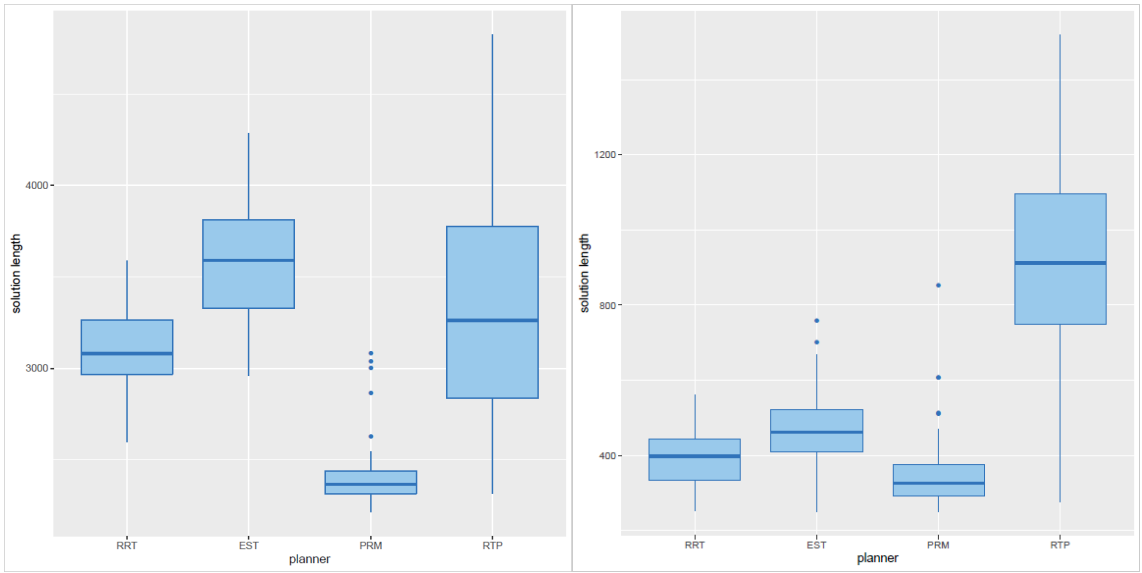

    And based on many times testing, we found even though our planner can reach the goal, the solution path was **different** in each time.
  - The solution path is **not a smooth line**, from images above, we can see that the path usually takes a **sudden turn** into another direction. In real world test, the car might not be able to do this, therefore, the path is inappropriate and need to be improved.

- 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).
  - We benchmarked our implementation of RTP with PRM, EST, and RRT planners in Cubicles and Twistycool scenarios. Then generated plot graph from Planner Arena.
    - For Cubicles, we set runtime limit to ***50*** seconds, memory limit to ***10000*** mb as default, and performed ***50*** independent runs. The total running time is about 40 minutes.
    - For Twistycool, we set runtime limit to ***100*** seconds, memory limit to ***10000*** mb as default, and performed ***50*** independent runs, The total running time is about 2 hours.
    - We could see that RTP spent much more time than the other three planners. The left plot is Cubicles and the right plot it Twistycool scenario.



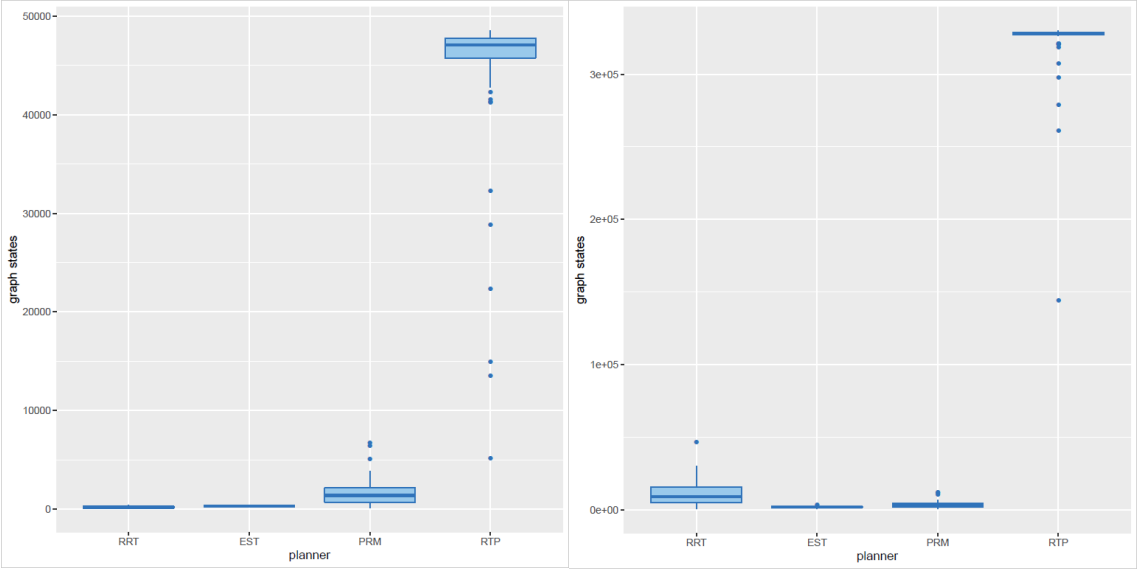

*Planner time consumption for different scenarios, Cubicles vs Twistycool*

- Path length: RTP planner has longer path of found solutions, especially for the Twistycool scenario.




*Planner path length for different scenarios, Cubicles vs Twistycool*

- Graph states: As we can see RTP generates more states than other three planners.




*Planner graph states for different scenarios, Cubicles vs Twistycool*

- Since we have the generated SQLite database file, then we ran some SQL statements to query the average performance of four planners so that we could see their average performance in numbers.

| *Cubicles* | RRT | EST | PRM | RTP |
|---|---|---|---|---|
| Time | 0.4171085 | 0.83882174 | 5.5891633 | 46.295575 |
| Path length | 3118.5812 | 3594.2714 | 2415.372 | 3313.6166 |
| Graph states | 223.42 | 338.18 | 1742.24 | 43394.22 |

*Average time, path length and graph states for planners in Cubicles*

| *Twistycool* | RRT | EST | PRM | RTP |
|---|---|---|---|---|
| Time | 5.75788264 | 3.4894643 | 8.5269688 | 98.082666 |
| Path length | 389.83926 | 465.6366 | 354.91036 | 899.7398 |
| Graph states | 11734.84 | 1899.86 | 3546.46 | 320522.78 |

*Average time, path length and graph states for planners in Twistycool*

- By the plot images and the average performances list above, we could know that RTP is less efficiency compare with other three planners. It is very intuitive and straightforward to see this difference from the speed of the progress bar when we ran the benchmark. The RTP took much longer time than the other three planners.

➢ 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, for students who completed the project in pairs, describe your individual contribution to the project.

- ✧ Exercise 1: Difficulty: 6, Spent hours: 6 hours
- ✧ Exercise 2: Difficulty: 5, Spent hours: 3 hours
- ✧ Exercise 3: Difficulty: 5, Spent hours: 7 hours

The hardest part of this project is understanding the logic behind the code of OMPL library, its systematic

and each part is connected with several others. Even if you just want to read one coding file, you might need to read more than 10 other files due to the cross-file function calls.

**Contributions:**

- ✧ Hao Ding:

  Exercise 1: Implemented the RTP planner.

  Exercise 2: Designed the testing cases and visualized the results.

- ✧ Haoran Liang:

  Exercise 1: Implemented the RTP planner.

  Exercise 3: Benchmarked implementation of RTP with other three planners in two scenarios.