COMP550 - Project 3

Anja Conev - ac121
Dejan Grubisic - dx4

October 17, 2019

1. **A succinct statement of the problem that we have solved:**

   For this project our task was to **implement a RTP (Random Tree Planner)** in the OMPL environment. Using the logic for Planner construction in OMPL we implemented the simple algorithm described in the instructions.
   Next, we had to **plan for a point and a square robot** with this planner in two different environments. We have created two different environments for our robots. We have instantiated our planner and set all the required parameters for planning. We have written a short Python script for the visualization of the results.
   Finally, we have **benchmarked the results obtained with our simple RTP** against PRM, EST, and RRT planners.

2. **A short description of the robots (their geometry) and configuration spaces we tested in exercise 2:**

   (a) **Point robot** - this robot has two degrees of freedom and it's configuration is uniquely defined by two parameters - x and y coordinates of its position in our 2D Euclidian workspace. Its configuration space is exactly the same as the workspace $C = R^2$. The obstacles in the configuration space look the same as the obstacles in the workspace which makes our problem easy - we are just checking for collision in the workspace.

   (b) **Square robot** - it translates and rotates in the R^2 workspace. Its position is defined by x and y coordinates of its center and the angle of rotation θ. This robot has three degrees of freedom. Its configuration space is defined as: $C = R^2 \times S^1 \cong R^2 \times SO(2) \cong SE(2)$. Collision checking for this robot is done in the workspace and sampling is done in the configuration space.

3. **Images of our environments and a description of the start-goal queries we tested in exercise 2:**

   We have constructed two different environments for square and point robot and different starting positions for the square and point robot. All those configurations are shown in the following figures. Green rectangles are the obstacles, the starting state is the red position of the robot and the blue position is the goal position.

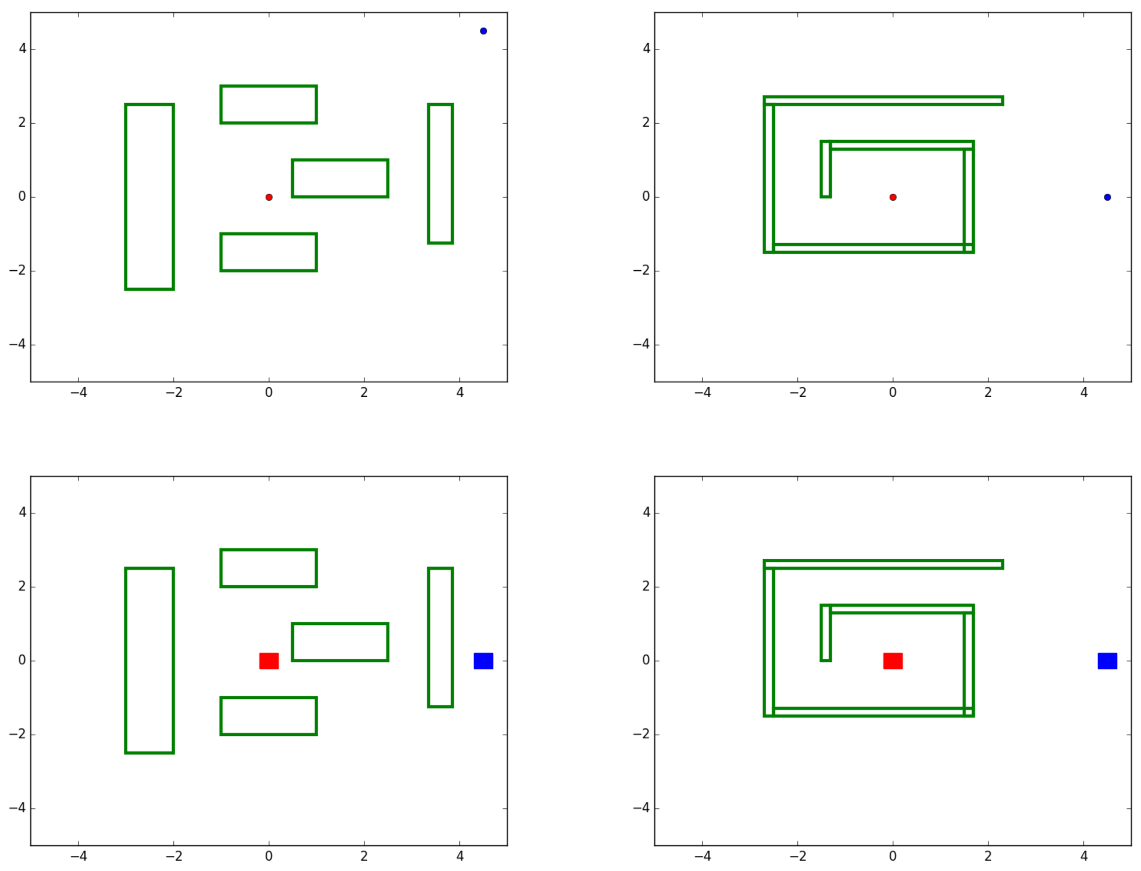


Figure 1: environments and start and goal configurations for a point robot - above, and square robot - below (obstacles - green, start - red, goal - blue)

4. **Images of paths generated by RTP in the environments we tested in exercise 2:**

   In Figure 2 we present the performance of our RTP visualized by our Python script, for the environments described in figures above. We present the performance of our robots in two different runs. For the square robot we did the interpolation of the resulting path, so we plot all the intermediate positions too. You can see that in some cases our objects get very close to the obstacles. Since the collision checking of the path is done with a certain resolution it is possible that a collision has been missed due to this resolution.

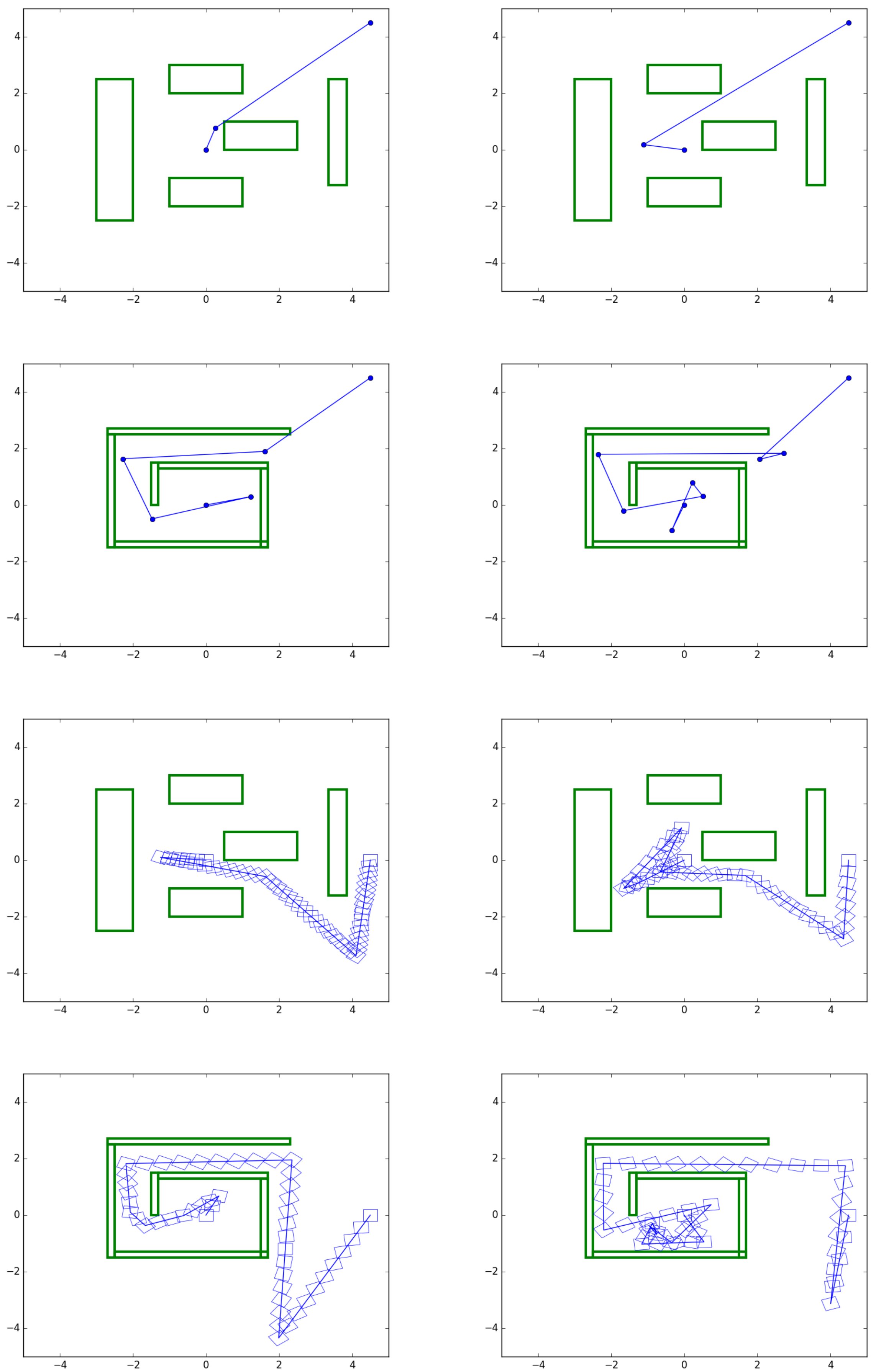


Figure 2: solutions for point and square robot in two environments in two different runs

5. **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?**

   Implementing the RTP planner was an interesting experience. We have familiarized ourselves with the OMPL library. The benefit of separation of the planner from the problem instance became more clear. We were able to use exactly the same planner on two different problems with completely different configuration spaces. We have implemented the RTP by looking at the implementation of RRT and figuring out which parts should be implemented differently for our, simpler planner. Important part of the work was visualization, which helped us see how our new planner is behaving. To test our implementation we have constructed two environments. The first one has five rectangular obstacles and a lot of free space around them. The second one is shaped like a rectangular spiral and is more challenging to solve. In the first environment our planner easily solved both problems. We can see that the path it finds is not optimal in any way - this is due to the fact that our RTP finds all states randomly and does not do any optimization while finding the path. The second environment was more challenging, and for the square problem our planner solved the problem significantly slower, but it still managed to solve it in 1 second for every run we did. It is important to stress that our RTP chooses a state from the existing tree at random and also chooses a new state for collision checking from the conformation space at random - this means that no clever optimization of the search is done - this is also visible in the generated paths which all seem suboptimal.

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

   The benchmark for RTP planner was developed based on the implementation of the existing example benchmark *SE3RigidBodyPlanningBenchmark.cpp*. For benchmarking we used *Cubicless* and *Twistycool* environments. 3D models of those environments and start and goal configurations are shown on figure 3. Shape and start and goal configurations of robots in the workspace are represented with orange and red color respectively. The figure also contains the visualization of the states generated by RRT planner with time limited to 20s.

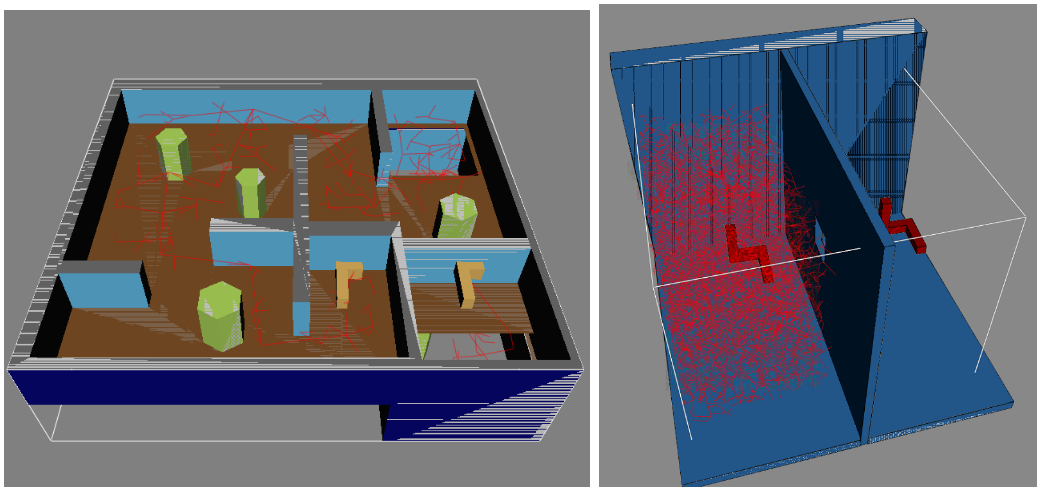
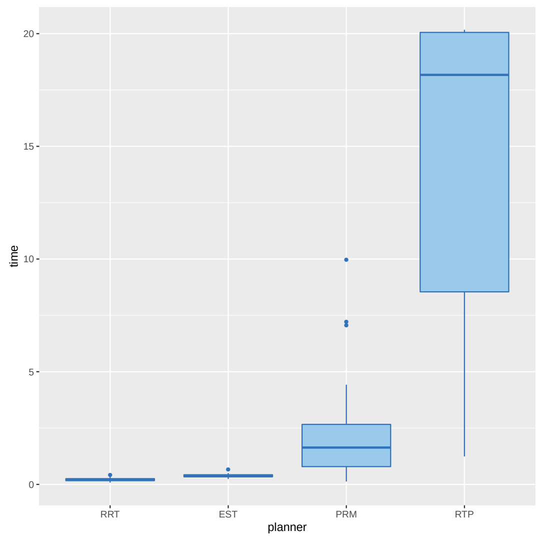
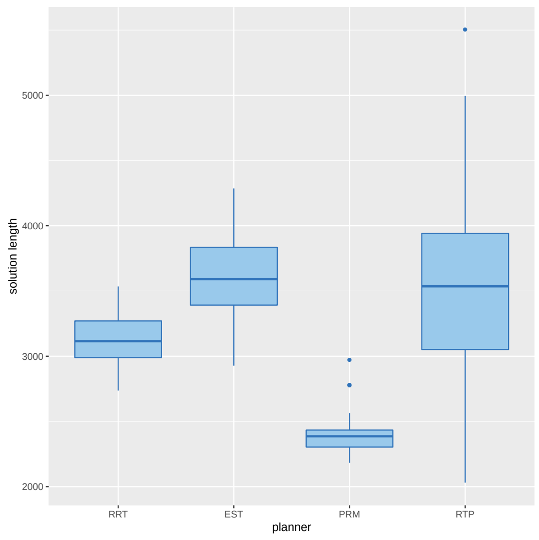


Figure 3: Configurations Cubicles and Twistycool with graph of states

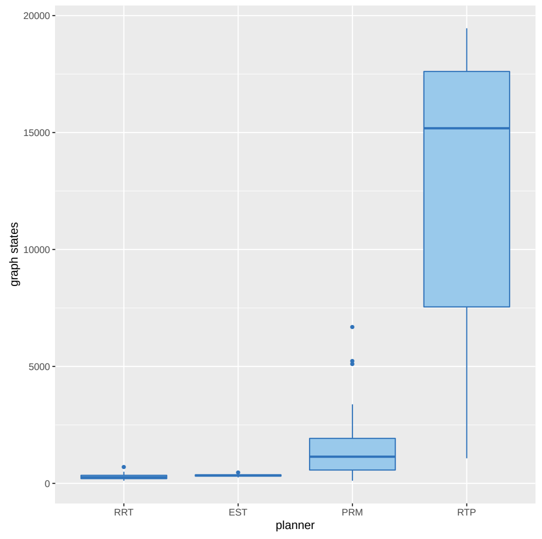
In the first benchmark we compare planers PRM, EST, RRT with our implementation of RTP planner on *Cubicles* configuration, with time limit per solution 20s and 50 runs. In figure 4 we present the results for the: time for finding the solution, length of the found path, number of states in constructed graph and ratio between exact and approximate solutions.



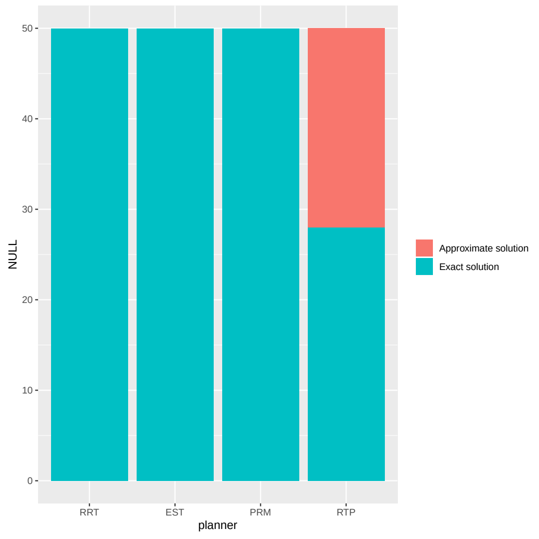
Execution time



Path length



Number of graph states

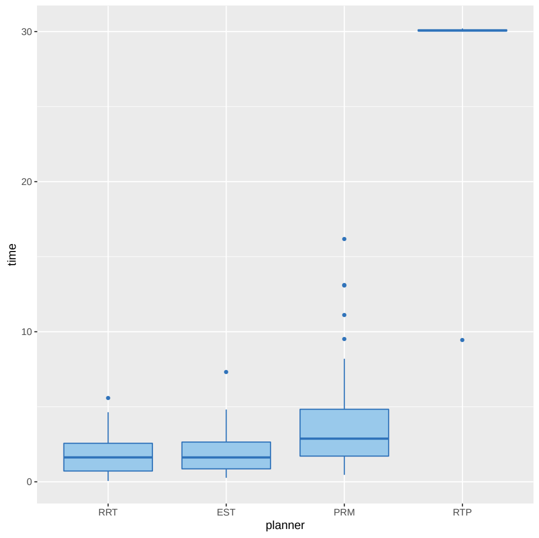


Ratio of exact and approximated solutions

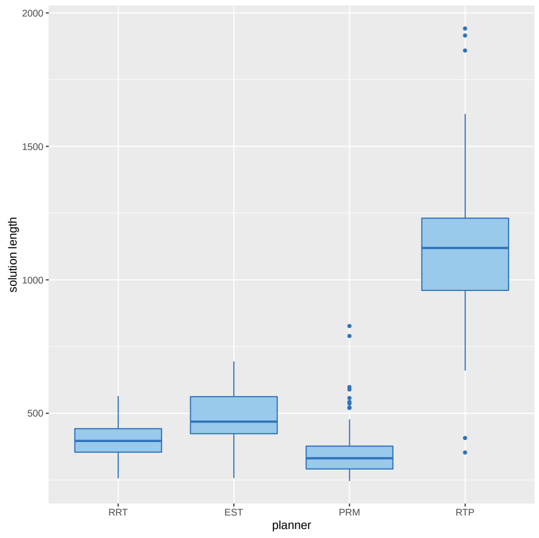
Figure 4: Benchmark for configuration. Cubicles

As we can see, execution time is considerably higher for RTP with median close to 20s what was time limit. This also explains why about 55% of solutions we got were exact, while 45% were approximate solutions. RTP planner also has the biggest variance in path length, because of its random nature. Number of generated states is more than 15 times bigger for RTP than in case of other planners, because there is no optimization of the path and the planning is done by RTP.

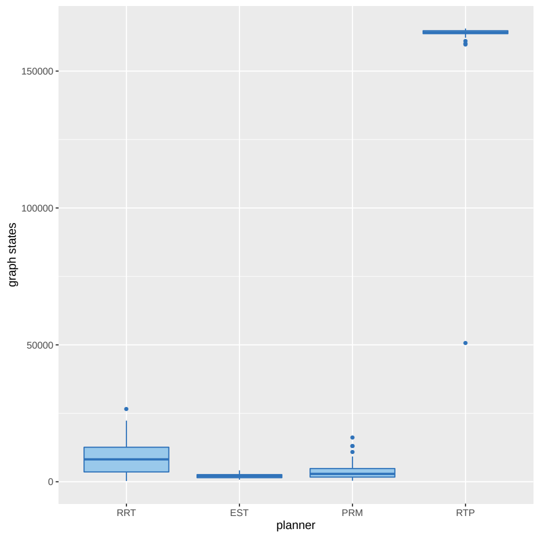
Second benchmark compares planners performance for *Twistycool* configuration. From figure 3 it can be seen that this problem is much harder to solve than the previous one, as the number of sampled states is much higher. For this problem our benchmark contains 50 runs with the time limit of 30s. The results are shown in figure 5.



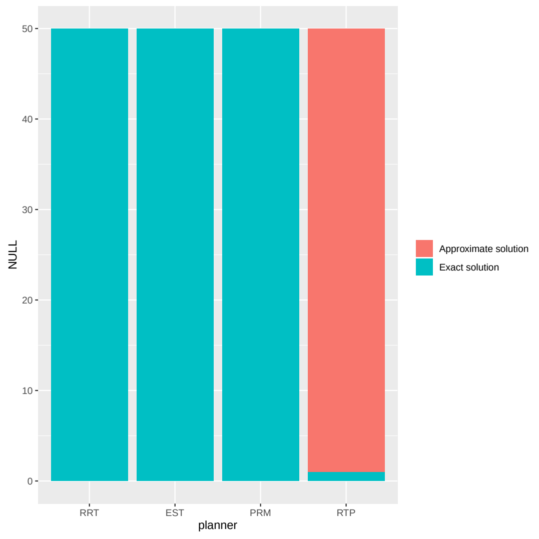
Execution time



Path length



Number of graph states



Ratio of exact and approximated solutions

Figure 5: Benchmark for configuration Twistycool

From execution time we can see that our planner hardly managed to find exact solution, that is also confirmed with ratio between exact and approximate solutions. RTP again has a bigger path length than other planners and the number of sampled states is more than 15 times higher than for other planers. Small variance in number of graph states is a consequence of time constrain, as majority of execution are terminated before finding exact solution.

Overall, the performance of our RTP planner is much worse than others in terms of all tested parameters: execution time, path length, number of sampled states, number of exact solutions found. This is a consequence of the fact that no optimization of planning is done by RTP and all sampling is done randomly.

7. **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.**

Project Exercises	Difficulty(1-10)	Time(hours)	Problems
1. Implement RTP	5	5	We had no problems
2. Compute valid motion plan	4	4	We had no problems
3. Benchmark	3	5	Not clear documentation

In this project both of authors worked on all parts together. However, Anja was engaged a little more in first two parts, while Dejan was more focused on the third part.