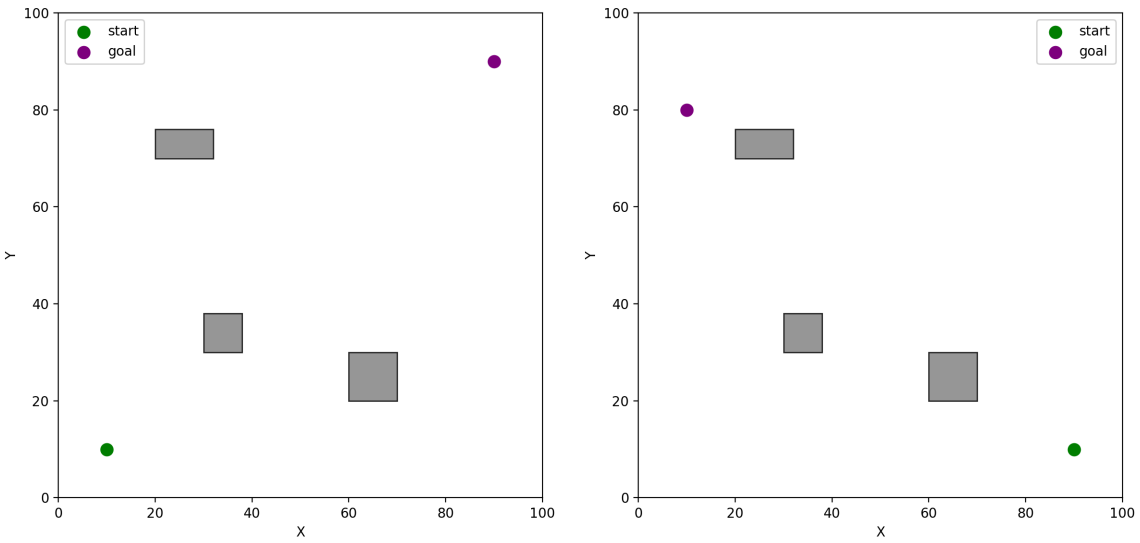
vc68 cl307 project 3 report

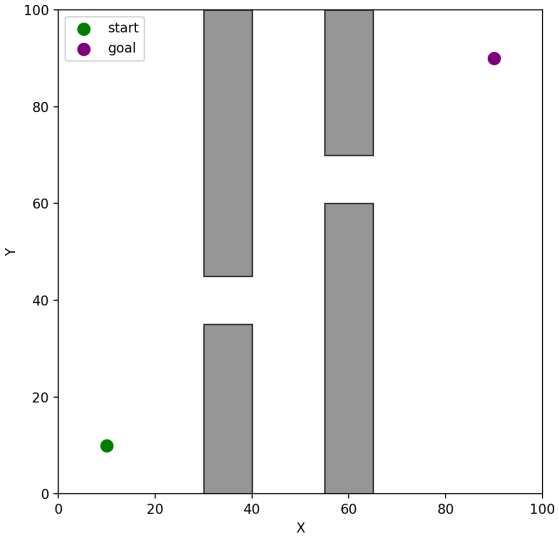
1. In this project, we aim to solve a path planning problem for point and square robots in two customized environments by implementing Random Tree Planning algorithm. Furthermore, we compare performance of four planners including RTP, RRT, RRT-Connect and PRM with metrics such as planning time, path smoothness, path length and path clearance.

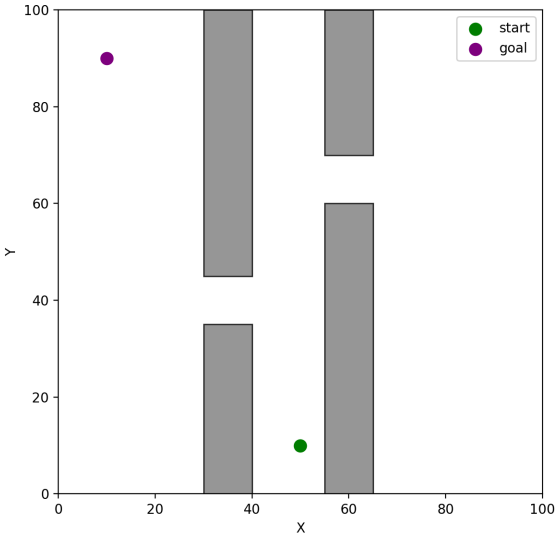
2. In exercise 2, we have two kinds of robots, a point robot whose configuration space is R ^ 2 in a 2D environment, and a square robot with side length = 5, whose configuration space is SE(2) in a 2D environment.
   We created two environments with rectangular obstacles. The limitation of planning time is sixty seconds and the visualization of two 2D environments we created are shown down below.

3. We test different pairs of start-goal points in those settings in exercise two.



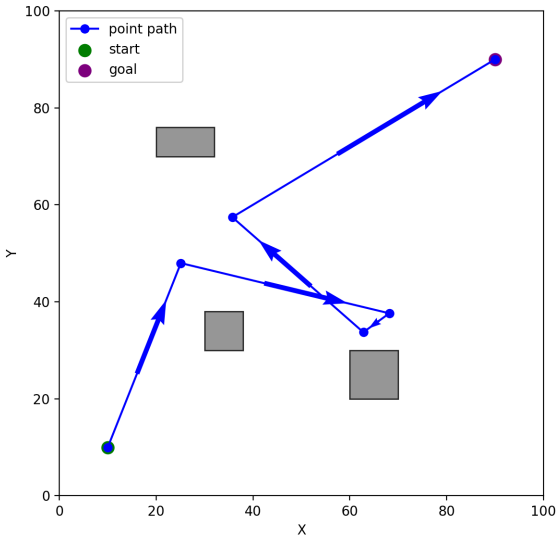
Left: Easy environment with (10, 10) as start and (90, 90) as goal.
Right: Easy environment with (90, 10) as start and (10, 80) as goal.

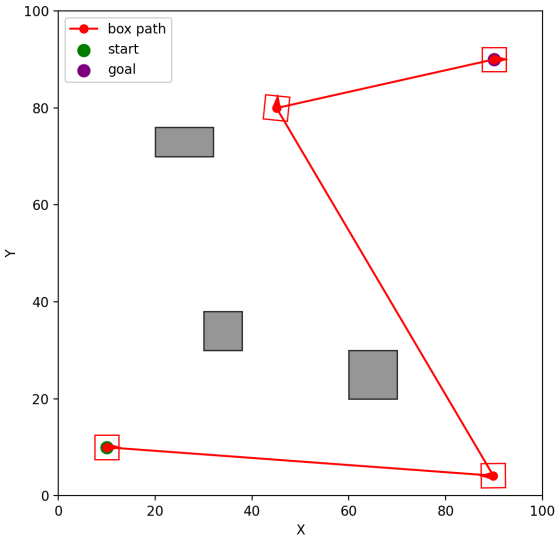


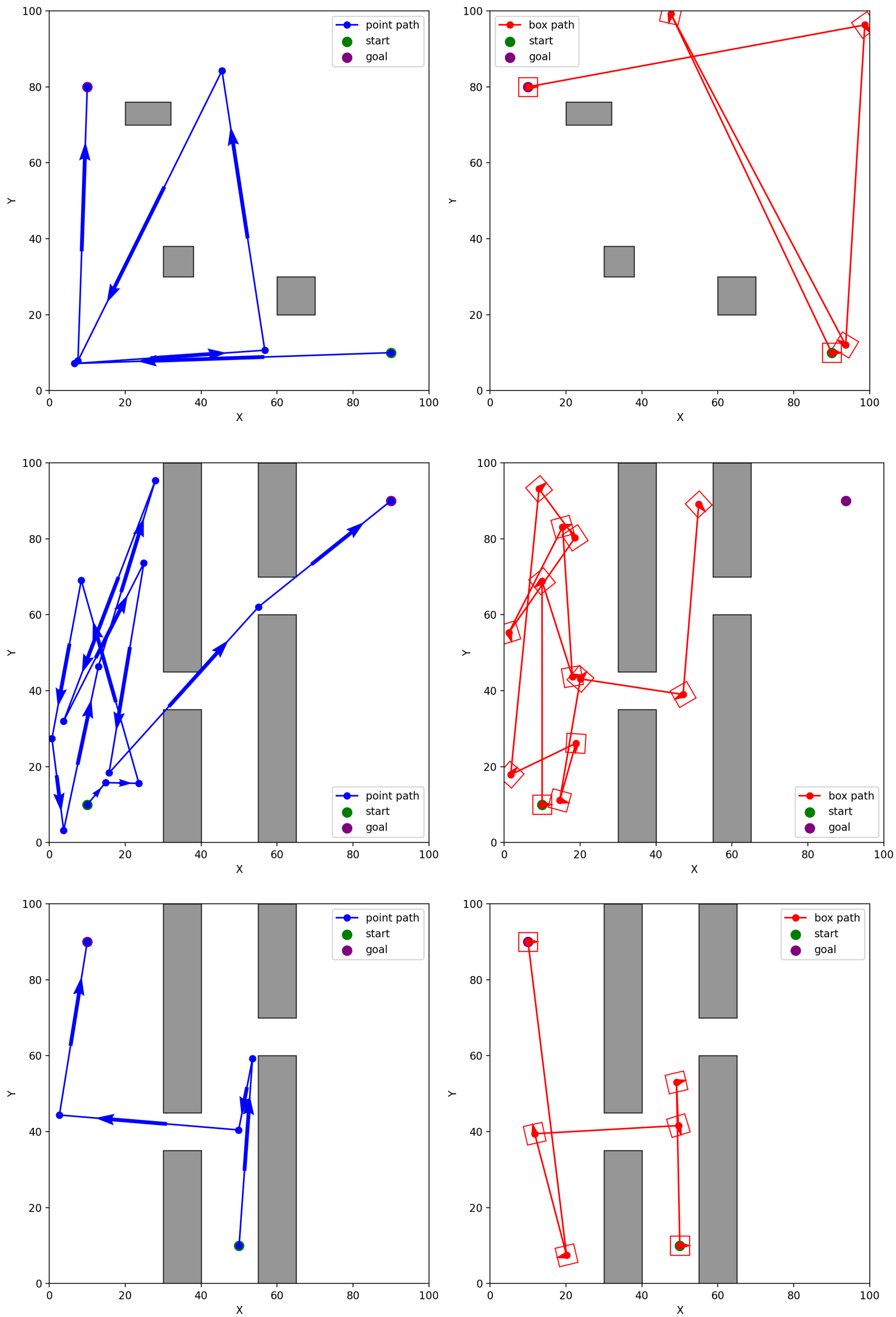


Left: Hard environment with small gaps between obstacles with (10, 10) as start and (90, 90) as goal.
Right Hard environment with (50, 10) as start and (10, 90) as goal

4. The visualization of planning paths are down below.



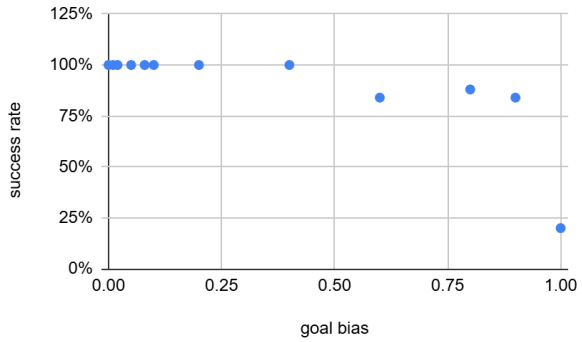


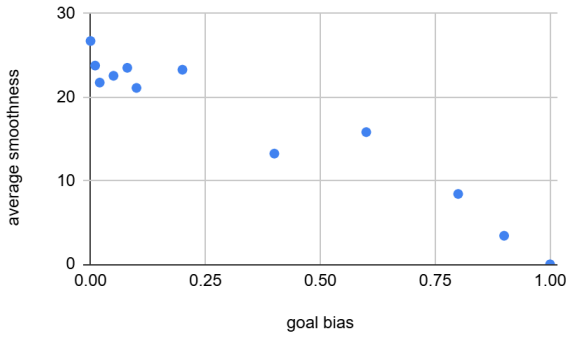

point path
start
goal
box path
start
goal
X
Y

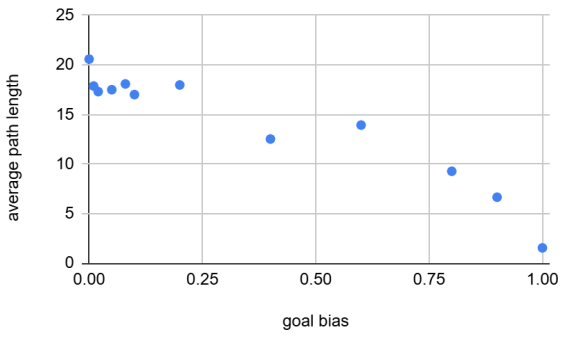
5. During the implementation we can know that RTP is actually an elementary route planning method that didn't leverage the closest states in the tree and randomly selected states in the tree to extend. As a result, the generated routes are winding and rugged and pass the same area repetitively although there's only one planner in a hard environment with (10, 10) as start and (90, 90) as goal returns an approximate state in sixty seconds (550k+ states but still can't reach). In summary, the performance of RTP is definitely not the best in terms of smoothness, path length, but it is an intuitive and easily-implemented algorithm.

6. In this part of the project, we tuned each planner by varying one parameter at a time while keeping all others fixed, allowing us to observe how changes in that parameter influence the planner's performance across the four evaluation metrics: planning time, path length, path smoothness, and clearance. Each parameter setting was evaluated across five different start–goal pairs, with five runs per pair.
   Link to experiment results: Project3_exp_table

   - RTP



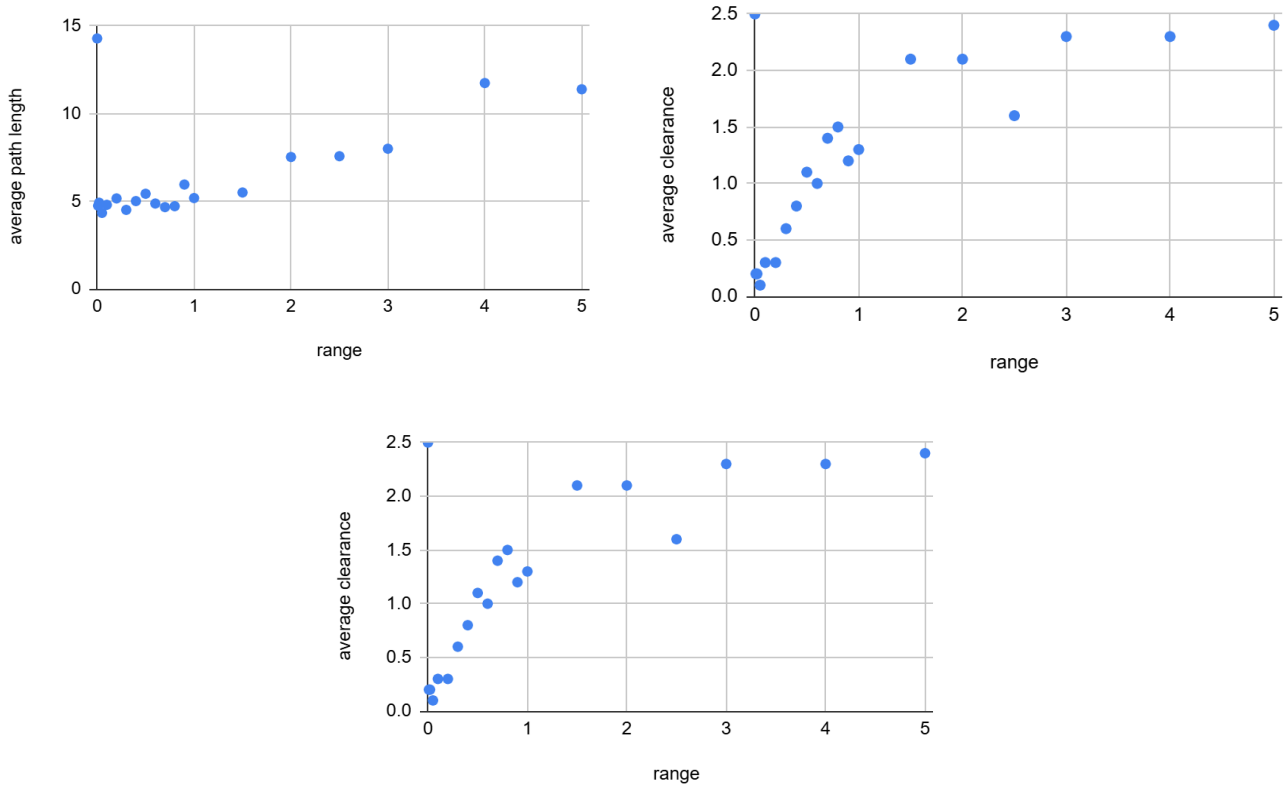




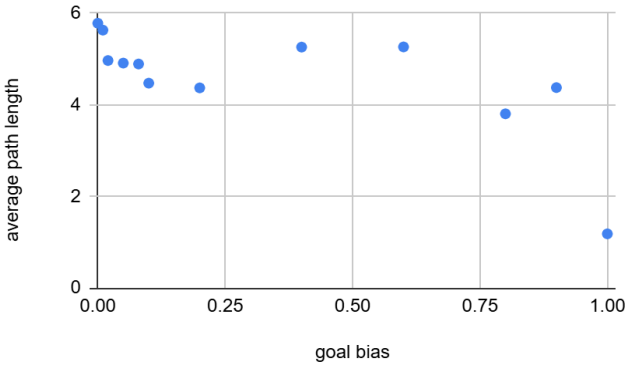
The results show that RTP maintained a 100% success rate for goal biases below 0.6, but it dropped beyond that point because the planner became too greedy and

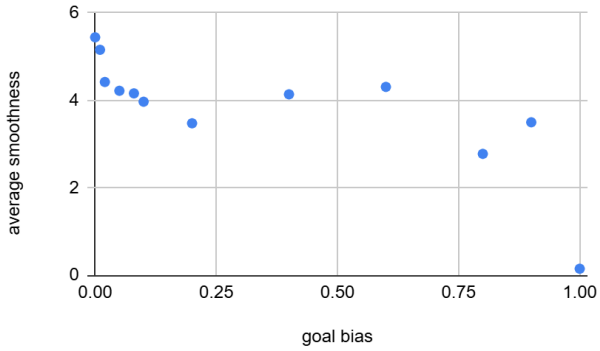
failed to explore alternative routes. Path length and smoothness improved as goal bias increased since the planner grew more directly toward the goal, but this also reduced robustness in complex environments, leading to lower accuracy. Overall, we chose a balance between these metrics, selecting a goal bias of around 0.05 to maintain high success while improving path quality.

- RRT



When tuning RRT, we first varied the range parameter while keeping the goal bias fixed at 0.05. Small ranges produced the shortest and smoothest paths because the planner extended in smaller and more precise steps. As the range increased, path length and smoothness worsened since larger steps led to overshooting and less efficient connections. However, clearance tended to improve with larger ranges because the planner could step over narrow obstacles and avoid tight spaces. Based on these results, we selected a range of around 0.5 as the best balance.



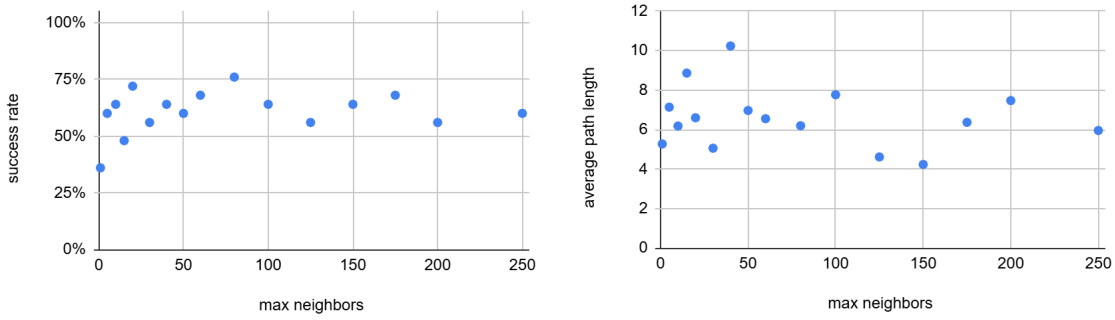


We then tuned goal bias while fixing the range at 0.5. Similar to RTP, path length and smoothness improved as goal bias increased because the tree grew more directly toward the goal. However, a too high goal bias reduced exploration and could harm success in complex environments. We therefore choose a goal bias of around 0.05 - 0.1 to achieve the best trade-off.

- RRT-Connect
  When tuning RRT-Connect, we varied the range parameter while keeping the goal bias fixed at 0.05, but the results showed no clear or consistent trends. The success rate stayed at 100% for all values, and while planning time, path length, and smoothness fluctuated slightly, there was no strong correlation with range. This suggests that RRT-Connect is relatively insensitive to the range parameter, likely due to its bidirectional growth strategy, which helps it connect to the goal efficiently regardless of individual step size.

  Similarly, when adjusting the goal bias (with range fixed at 0.5), we again observed little variation across the metrics. Success remained 100%, and changes in path length, smoothness, and clearance were minor and inconsistent. This indicates that goal bias has a much weaker effect on RRT-Connect compared to other planners like RTP and RRT. Overall, because of this robustness, we chose a range of about 0.5 and a goal bias of about 0.05 as our final parameter settings.

- PRM



PRM's success rate was generally lower, ranging from 36% to 76%, with the best results achieved around 80 neighbors. Increasing the number of neighbors improved the connectivity of the roadmap, which helped the planner find paths more reliably, but too many neighbors also increased computational cost and sometimes led to longer or less efficient paths due to unnecessary connections. Path length, smoothness, and planning time reached their lowest point at around 125 neighbors, indicating that more connections can improve path quality up to a certain point. Overall, we found that setting the maximum number of neighbors to around 100 provided the best balance between success rate, planning time, and path quality.

- Best Parameter Settings for Each Metric (Successful rate = 100%)
  According to the experiment result, we choose RRT-C for planning time due to its bidirectional speedup, and for clearance because its rapid tree growth often favors wider, more open regions of the space. As for path length and smoothness, we choose RRT because it connects each new sample to the closest existing node (which may be the goal when goal bias is applied). This incremental and locally optimized growth helps build more direct and efficient paths, resulting in shorter trajectories and smoother motion overall.

    - Average planning time: 1.9025 s – achieved by RRT-Connect with range = 0.5 and goal bias = 0.1
    - Average path length: 4.369 – achieved by RRT with range = 0.5 and goal bias = 0.2
    - Average smoothness: 3.475 – achieved by RRT with range = 0.5 and goal bias = 0.2
    - Average clearance: 2.5 – achieved by RRT-Connect with range = 0.5 and goal bias = 0.02

7. Exercise 1: diff: 9, time: 7 hrs; Exercise 2: diff: 8, time: 2 hrs; Exercise 3: diff: 7, time: 7 hrs. The hardest part of the project is to implement the RTP.h and RTP.cpp in exercise 1, we spent lots of time to understand the architecture in existing OMPL planners such as RRTs to implement RTP ideas, especially the solve function.
   Individual contribution: vc68: Exercise1 solve function, Exercise 2, Exercise 3 experiments design; cl307: Exercise 1 RTP.h, RTP.cpp, Exercise 3 experiment design and analysis.