Homework 2 – Planning for a high-DOF planar arm

Akshit Mishra (akshitm)

November 1, 2020

Abstract

For this homework, the aim was to implement a range of sampling-based planners for the arm to move from its start joint angles to goal joint angles. The planners needed to provide a path that was collision-free. The algorithms implemented in this assignment are: Probabilistic Road Map (PRM), Rapidly Exploring Random Trees (RRTs), Rapidly Exploring Random Trees Connect (RRT-Connect), Rapidly Exploring Random Trees Star (RRT*).

Approach: For the results shown below, the following methodology was employed:

- Randomly generate 25 **valid** start and goal configurations pair by varying one of the links' orientation randomly.
- For each valid and random arm configuration pair, call all the planners one after the other on map2 to juxtapose their performances.
- The final performance parameters were recorded for each run to generate the overall comparison table shown below in 1
    1. The path quality for each run was computed as the sum of euclidean distance in the C-space for the final path.
- Additionally, five runs were chosen for each planner to show variation in performance with change in start and goal positions.
- **The specific methodology for each planner in described in its respective section after reporting the overall results in table 1.

1 Results

Planner	**Average Planning Time** (s)	**Success Rate** (%)	**Average Number of vertices generated**	**Average Path Quality**
PRM	2.75	100	10002	24.20
RRT	0.25	87.50	350	15.70
RRT Connect	1.75	91.30	3800	14.11
RRT*	0.70	100	2224	12.24

Table 1: Summary of PRM Test Results (Map 2)

1.1 Overall Results Discussion

After running all the algorithms with random start and goal configurations, it was clear that each planner had its own set of pros and cons. For instance, PRM was a more robust approach in terms of handling all different kinds of start and goal configurations, but took more time to generate a plan than the RRT-based planners. Moreover, the memory used by PRM was also much higher than all the other planners. In terms of only execution time, RRT and RRT* were the best approaches. Although RRT was usually faster than RRT*, the path quality was significantly better for RRT* and the end position of the arm was usually much closer to the goal for RRT*. Furthermore, RRT* was much more robust for changes in start and goal configurations, and was able to generate a path for all cases in less than 5 seconds but same was not the case for RRT. In terms of run-time memory, PRM was the worst and RRT, with its set of other flaws, turned out to be the best. In terms of path quality, RRT* was easily the best approach and it was also the most robust approach out of the sampling-based planners. Overall, **RRT*** was easily the best approach out of all the planners tested on Map 2. It generated collision-free plans for most cases in less than a second without nearly occupying the same memory as PRM. So it provided the robustness of PRM while improving the path quality, run-time speed and memory allocation. It was also much more reliable than other RRT-based counterparts as the success rate of RRT* was noticeably higher than RRT and RRT-connect. Moreover, the path quality for RRT* was also significantly better in most cases.
The only area where RRT* could improve was the average number of vertices in the graph. In comparison to RRT, RRT* connect was almost an order of magnitude worse in terms of the size of graph created. This issue can be tackled to a certain extent by choosing more efficient data structures or using better sampling strategies so that a path is found without having to create a lot of vertices in the graph.

1.2 Extra Credit: Additional Statistics

The table below summarizes the variances observed for each of the planners when the algorithms are run with similar start and goal configurations.

Planner	**Variance of Planning Time**	**Variance of Success Rate** (%)	**Variance of Number of vertices generated**	**Variance of Path Quality**
PRM	0.045	–	–	35.30
RRT	51.35	0.122	350	151.62
RRT Connect	0.22	0.269	60230	12.68
RRT*	6.52	0.20	48008673.53	7.37

Table 2: Summary of PRM Test Results (Map 2)

1.3 Probabilistic Road Map (PRM)

1.3.1 Approach

- The algorithm was designed to generate a roadmap with 1000 vertices for all cases. Thus, the size of the graph was fixed for all the scenarios.
- An exhaustive search was used to find the closest K neighbors within a radius of the current vertex.

- If the new node didn't have an existing path to any of the neighbors, an edge was added.
- **Search Methodology:** ***Weighted A Star*** was used with a weight of 5 to search for the shortest path in the constructed roadmap for map2. [Lik20]

write about fixed graph size average time for **20** iterations

1.3.2 PRM Results Summary

Start Configuration	Goal Configuration	Planning Time(s)	Path Length	Graph Size
[1.0496 1.4529 1.4751 1.5020 1.0914]	[2.5119 1.6328 5.0270 2.7107 5.7218]	2.52	122	10002
[1.3034 1.2240 1.5185 1.2405 1.1054]	[4.9025 2.4488 1.5186 2.5379 0.6060]	2.70	80	10002
[1.5405 1.5478 1.3484 1.0785 1.1701]	[2.2190 5.1597 0.0968 0.2703 1.0618]	2.69	141	10002
[1.4303 1.3864 1.2833 1.3336 1.2024]	[4.6790 1.1872 4.3151 1.1530 2.3153]	3.04	71	10002
[1.4557 1.0897 1.5338 1.4534 1.3021]	[2.7386 2.8072 1.9249 3.1951 3.2093]	3.23	84	10002

Table 3: Summary of PRM Test Results (Map 2)

1.3.3 Results Discussion: PRM

1. The graph construction was the most expensive part of this approach.
2. Due to computationally expensive graph-creation process, PRM was order of a magnitude slower than the RRT-based approaches.
3. For comparable number of iterations, PRM was more robust in terms of finding a plan even with more collision and complicated maneuvers than RRT and RRT-Connect.
4. The graph size of PRM was almost an order of magnitude larger than RRT and RRT-Connect.
5. The advantage of more dense graphs with PRM was that the final plan was much smoother and less sparser than all the RRT-based planners.
6. The search radius for these runs was: $PI/4$

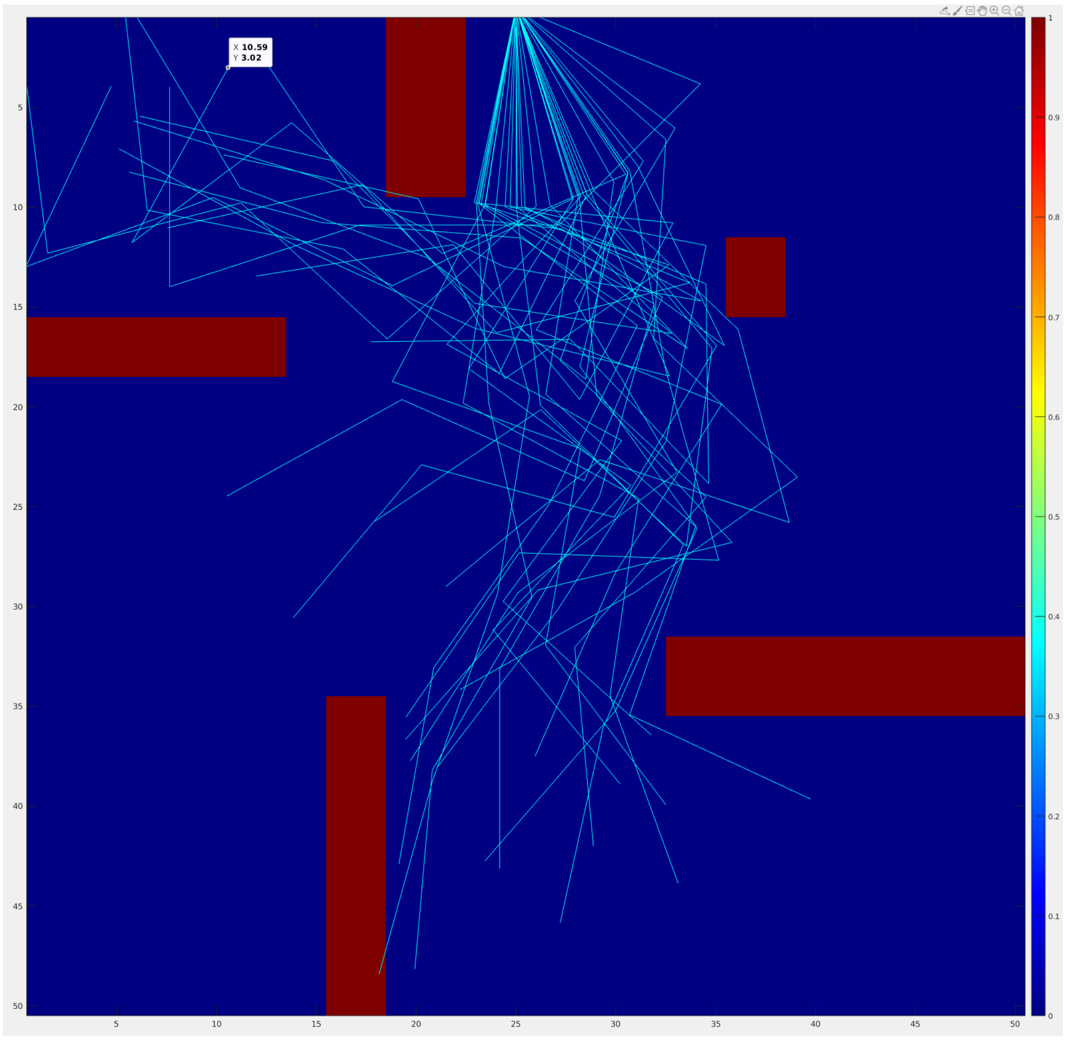

Figure 1: Sample from PRM planner on Map2 with start configuration [pi/1.8 pi/6 pi/1.7 pi/1.5 pi/2] and goal configuration [pi/8 3*pi/4 pi 0.9*pi 1.5*pi]

1.4 Rapidly Exploring Random Trees (RRT)

1.4.1 Approach

- The start configuration was added to the graph
- Goal Bias was used, where 1% of the points were sampled closer to the goal.
- The closest neighbor was found using an exhaustive search over the graph.
- The nearest node was steered to the randomly sampled point by a distance of epsilon ($PI/4$).
- Process is repeated until the goal region is sampled, and added to the graph
- The radius of the goal region in this case was $PI/8$.

Start Configuration	Goal Configuration	Planning Time(s)	Path Length	Total Vertices
[pi/1.8 pi/6 pi/1.7 pi/1.5 pi/2]	[pi/8 3*pi/4 pi 0.9*pi 1.5*pi]	0.22	13	1172
[pi/**1.70** pi/6 pi/1.5 pi/**1.5** pi/2]	[pi/**7.8** 3*pi/4 pi 0.9*pi 1.5*pi]	0.05	17	202
[pi/**1.75** pi/6 pi/1.5 pi/1.5 pi/2]	[pi/8 3*pi/4 pi 0.9*pi 1.5*pi]	0.13	15	302
[pi/1.70 pi/6 pi/1.5 pi/1.5 pi/**1.8**]	[pi/8 3*pi/4 pi 0.9*pi **1.25***pi]	0.55	13	302
[pi/1.7 pi/6 pi/1.5 pi/**1.65** pi/1.8]	[pi/8 3*pi/4 pi 0.9*pi 1.5*pi]	0.11	15	402

Table 4: Summary of RRT Test Results (Map 2)

1.4.2 RRT Results Summary

1.4.3 Results Discussion: RRT

1. The graph representation for RRT was much sparser than PRM.
2. The generated plans were also sparser due to smaller graphs. For this reason, if the algorithm was to be implemented on the actual arm, the local planner would need to be more sophisticated than it would have been for PRM.
3. RRT was an order of magnitude faster than PRM for finding a plan from start to goal.
4. The final position had more deviation from the desired goal configuration as RRT planned only until a vertex from "goal region" was added.

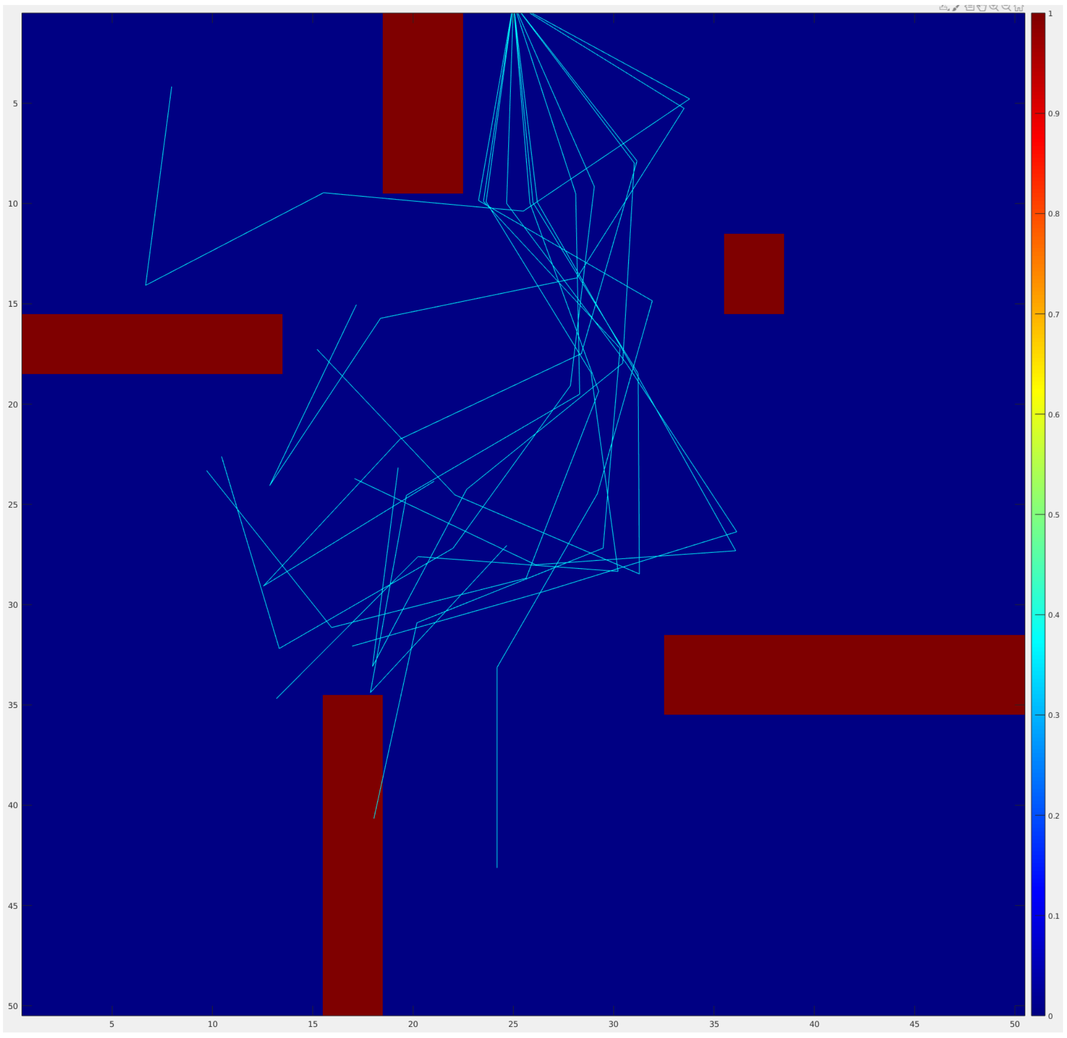
Figure 2: Sample from RRT planner on Map2 with start configuration [pi/1.8 pi/6 pi/1.7 pi/1.5 pi/2] and goal configuration [pi/8 3*pi/4 pi 0.9*pi 1.5*pi]

1.5 RRT Connect

1.5.1 Approach

Start Configuration	Goal Configuration	Planning Time(s)	Path Length	Total Vertices
[pi/1.8 pi/6 pi/1.7 pi/1.5 pi/2]	[pi/8 3*pi/4 pi 0.9*pi 1.5*pi]	1.02	12	3550
[pi/**1.70** pi/6 pi/1.5 pi/**1.5** pi/2]	[pi/**7.8** 3*pi/4 pi 0.9*pi 1.5*pi]	2.32	22	5030
[pi/**1.75** pi/6 pi/1.5 pi/1.5 pi/2]	[pi/8 3*pi/4 pi 0.9*pi 1.5*pi]	2.22	16	5665
[pi/1.70 pi/6 pi/1.5 pi/1.5 pi/**1.8**]	[pi/8 3*pi/4 pi 0.9*pi **1.25***pi]	1.28	15	5020
[pi/1.7 pi/6 pi/1.5 pi/**1.65** pi/1.8]	[pi/8 3*pi/4 pi 0.9*pi 1.5*pi]	0.72	23	2284

Table 5: Summary of RRT Connect Test Results (Map 2)

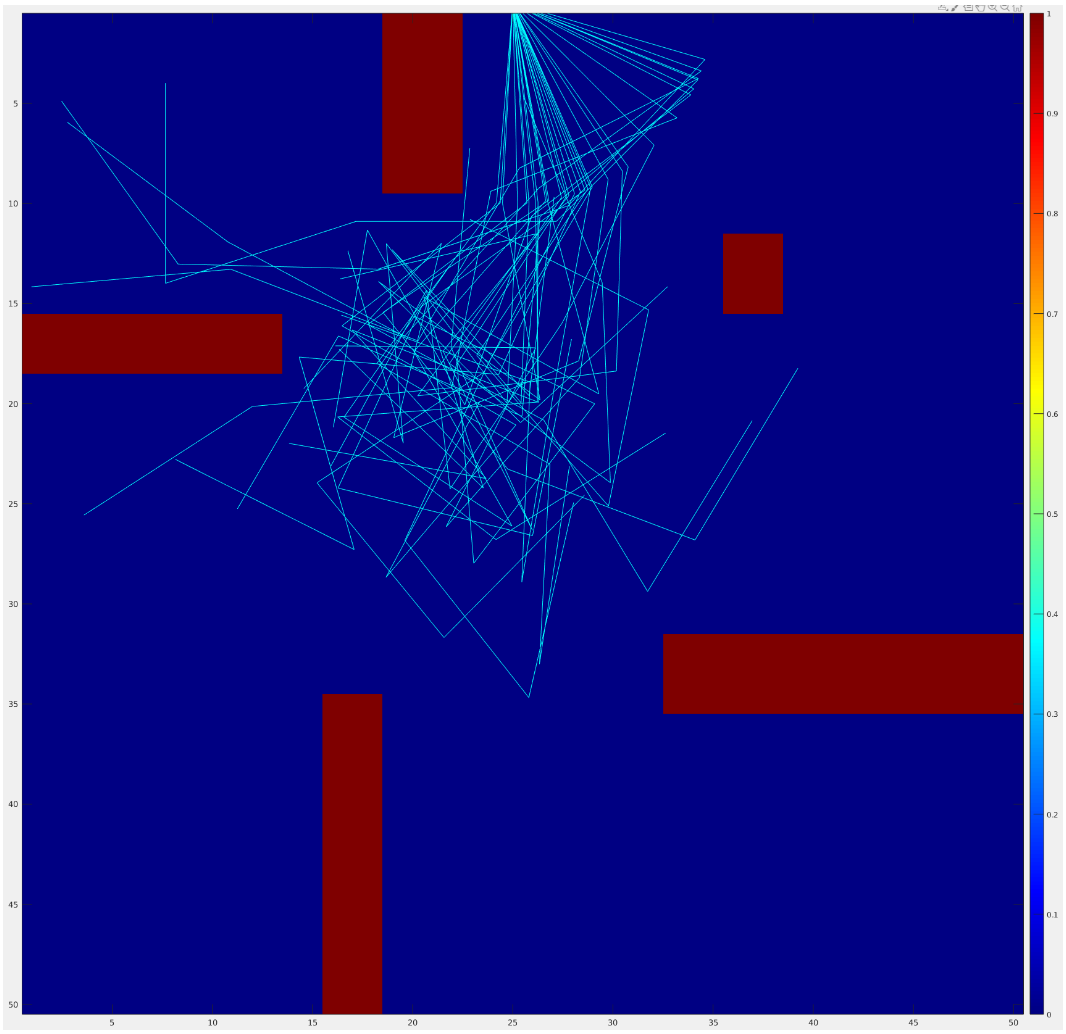
Figure 3: Sample from RRT Connect planner on Map2 with start configuration [pi/1.8 pi/6 pi/1.7 pi/1.5 pi/2] and goal configuration [pi/8 3*pi/4 pi 0.9*pi 1.5*pi]

2 RRT Star

Start Configuration	Goal Configuration	Planning Time(s)	Path Length	Total Vertices
[pi/1.8 pi/6 pi/1.7 pi/1.5 pi/2]	[pi/8 3*pi/4 pi 0.9*pi 1.5*pi]	0.10	13	962
[pi/**1.70** pi/6 pi/1.5 pi/**1.5** pi/2]	[pi/**7.8** 3*pi/4 pi 0.9*pi 1.5*pi]	0.92	25	2582
[pi/**1.75** pi/6 pi/1.5 pi/1.5 pi/2]	[pi/8 3*pi/4 pi 0.9*pi 1.5*pi]	0.67	18	962
[pi/1.70 pi/6 pi/1.5 pi/1.5 pi/**1.8**]	[pi/8 3*pi/4 pi 0.9*pi **1.25***pi]	0.75	13	3842
[pi/1.7 pi/6 pi/1.5 pi/**1.65** pi/1.8]	[pi/8 3*pi/4 pi 0.9*pi 1.5*pi]	0.40	11	1142

Table 6: Summary of RRT Star Test Results (Map 2)

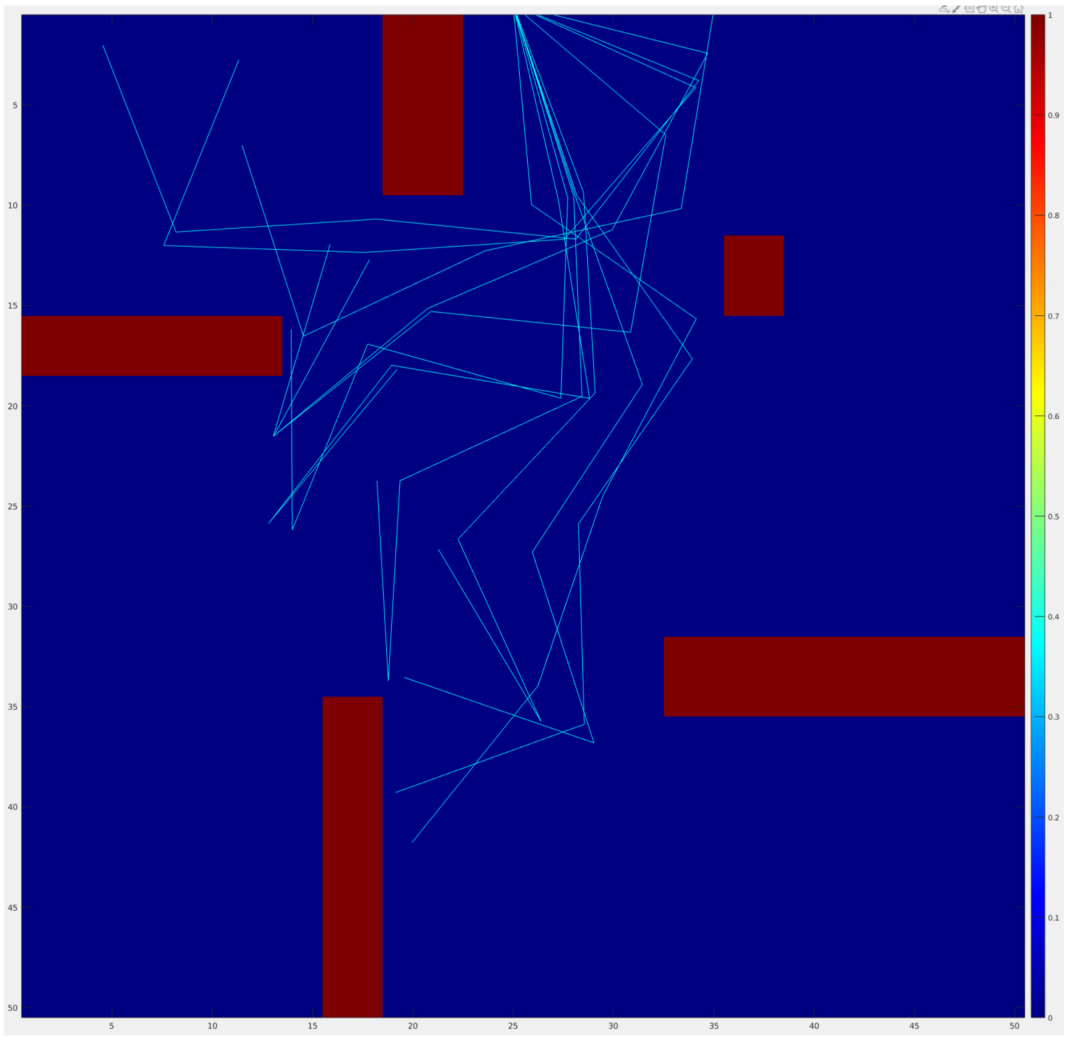
Figure 4: Sample from RRT Star planner on Map2 with start configuration [pi/1.8 pi/6 pi/1.7 pi/1.5 pi/2] and goal configuration [pi/8 3*pi/4 pi 0.9*pi 1.5*pi]

2.1 Use Case

```
mex planner.cpp
startQ = [pi/2 pi/4 pi/2 pi/4 pi/2];
goalQ = [pi/8 3*pi/4 pi 0.9*pi 1.5*pi];
planner_id = 0 % Choose the planner here
runtest('map2.txt',startQ, goalQ, planner_id);
```

Files required to run: <planner.cpp>
****In case one of the planners seems to take significantly longer than 5 seconds, please terminate that run, and run the program again without changing the configuration. This behavior is somewhat related to random nature of sampling-based approaches.**

References

[Lik20] Maxim Likhachev. “Interleaving Planning and Execution: Anytime and Incremental A*”. In: (2020). https://www.cs.cmu.edu/~maxim/classes/robotplanning_grad/lectures/execanytimeincsearch_16782_fall20.pdf.