HOMEWORK 2
Robot Autonomy

Naman Kumar, Jaiwei Lui and Sarah Betzig

Q1.

Iteration	Path Length	Plan Time (sec)	Vertices
1.	5.6	6.56	6
2.	9.5	2.6	6
3.	6.64	2	6
4.	7.3	1.87	7
5.	7.2	1.2	7
6.	6.89	2.5	9
7.	12.5	5.6	11
8.	8	1.96	8
9.	7.29	1.26	8
10.	5.7	1.35	4

Average Path Length	Average Plan Time	Average Number of Vertices
7.6	2.69 sec	7

The goal sampling probability of 0.1 was used.
The video is attached for the 2D Configuration space. The name of the video is Simple_RRT.mp4

Q2.

Iteration	Path Length	Plan Time (sec)	Vertices
1.	6.3	0.7	6
2.	6.4	1.96	11
3.	7.7	1.1	8
4.	6.7	0.5	6
5.	6.13	3.8	7
6.	10.6	3.1	11
7.	10	2.4	8
8.	7.8	0.95	8
9.	9	0.9	8
10.	8.5	4.4	11

Average Path Length	Average Plan Time	Average Number of Vertices
7.9	1.9 sec	8

1. In the Bidirectional version of RRT, we grow two trees, one from the starting point and the other from the goal and whenever the two trees connect, we have a path from the start config to the goal config.
2. In this case, both trees are moving towards each other as compared to the normal RRT where we had only one tree from the starting point.
3. Therefore, Bidirectional RRT is faster than the normal RRT, that is Bidirectional RRT takes less plan time as compared to normal RRT. Also, Bidirectional RRT has less average path length and number of vertices.
4. But in general, the difference between Normal RRT and Bidirectional RRT depends on lot of factors like the goal position, obstacles and the environment. Bidirectional RRT is better if a goal is at a position which is very difficult to reach otherwise.

The video is attached for the Bidirectional RRT. The name of the video is Simple_BiRRT.mp4

Q3.

<u>RRT Planner for the WAM Arm</u>

Iteration	Path Length	Plan Time (sec)	Vertices
1.	22.8	68.6	15
2.	8.6	93.8	16
3.	39.3	107.4	21
4.	31.3	139.4	24
5.	21.4	41.7	14
6.	28.2	81.9	14
7.	22.7	69	15
8.	11.45	90.6	7
9.	14.54	51.2	7
10.	20.7	18.9	10

Average Path Length	Average Plan Time	Average Number of Vertices
22.1	76.2 sec	14

Bidirectional RRT Planner for the WAM Arm

Iteration	Path Length	Plan Time (sec)	Vertices
1.	7.08	1.33	6
2.	7.36	6.3	9
3.	14.26	17.04	11
4.	16.6	41.6	11
5.	13.6	7.32	9
6.	14.1	29.7	13
7.	18.9	41.3	15
8.	28.6	18.1	12
9.	22.1	44.2	14
10.	23.1	32.1	12

Average Path Length	Average Plan Time	Average Number of Vertices
16.5	23.9 sec	11

1. As can be seen from the tables above, all the three factors: Average Path Length, Average Plan Time and Average Number of Vertices decreases if Bidirectional RRT is used.
2. Bidirectional RRT is faster as compared to the normal RRT because here, we have two trees, one from the starting point and the other from the goal point and therefore they are able to form a path quickly.
3.It is definitely useful to use Bidirectional planning in the higher dimensional configuration space.

   a) In the simple case (2D Configuration space), normal RRT was very fast. Although, bidirectional RRT was faster than the normal RRT but difference was not big. Therefore, using bidirectional RRT in 2D Configuration space did not offer a huge advantage.

   b) But in higher dimensional configuration space, normal RRT is slow but Bidirectional RRT is quite fast in this space as can be seen from the average plan time for both RRTs.

   c) Therfore, it is more useful to use Bidirectional RRT in higher dimensional configuration space where normal RRT is slow but that also depends on the position of the obstacles. For example, if a goal is at a position which is very difficult to reach, then Bidirectional RRT will be really useful as it will be able to find the path efficiently wherease Normal RRT will face lot of difficulties in finding the path.

The videos for the WAM Arm using RRT Planner and RRTConnect Planner are attached. The name of the videos are Herb_RRT.mp4 and Herb_BiRRT.mp4

Q4.

WAM Arm

The cost function used was the Path Length between the starting point and the goal point. It was calculated by adding the length of the paths between consecutive vertices in the path from the starting configuration to the goal configuration.

Iteration	Path Cost (before Path Shortening)	Vertices (before Path Shortening)	Path Cost (after Path Shortening)	Vertices (after Path Shortening)
1.	40.1	24	31.6	14
2.	18.1	15	13.7	7
3.	15.6	10	8.1	4
4.	37.5	19	26.9	10
5.	28.1	15	18.8	6
6.	17.8	10	11.2	5
7.	20.7	15	15.2	7
8.	31.1	19	23.8	10
9.	10.5	11	8.4	5
10.	29.1	13	16.1	5

Average Reduction in Path Cost	Average Reduction in Vertices
7.5	7.8

The video is attached for Path Shortening. First, it traverses the path before Path Shortening and then it traverses the path after Path Shortening. The name of the video is Herb_Path_Shortening.mp4

Extra Credit:
The video is attached for the simple robot for Path Shortening. First, it traverses the path before Path shortening and then it traverses the path after Path shortening. The name of the video is Simple_Path_Shortening.mp4