Homework 3

16-662, Apr. 9, 2014

Sun Chunyang (chunyans), Steven Ford (stevenf), Vamshi Konduri (kvk)

Q1

We ran a Breadth-First Search, Depth-First Search, and A-Star Search algorithm on HERB in the 2D environment. The results can be seen in Tables 1, 2, and 3. A-Star consistently performs the best, returning shorter paths in a shorter amount of time.

Figures 1, 2, and 3 display sample plots of the paths found by HERB in the 2D environment using the 3 different search algorithms at a resolution of 0.1.

Refer to the file *2d_astar.ogv* for an example video of one of our A-Star search algorithm runs.

Table 1: 2D Breadth-First Search Results

Parameter	Res = 0.05	Res = 0.10	Res = 0.25
Path Length	5.450 m	5.500 m	5.250 m
Plan Time	284.464 s	16.487 s	1.795 s
Nodes in Path	110	56	22
Closed List Size	21078	5420	844

Table 2: 2D Depth-First Search Results

Parameter	Res = 0.05	Res = 0.10	Res = 0.25
Path Length	119.050 m	68.700 m	35.250 m
Plan Time	315.689 s	13.981 s	6.720 s
Nodes in Path	2382	688	142
Closed List Size	16991	4192	664

Table 3: 2D A-Star Search Results

Parameter	Res = 0.05	Res = 0.10	Res = 0.25
Path Length	5.450 m	5.500 m	5.250 m
Plan Time	117.335 s	11.606 s	0.629 s
Nodes in Path	110	56	22
Closed List Size	768	295	105

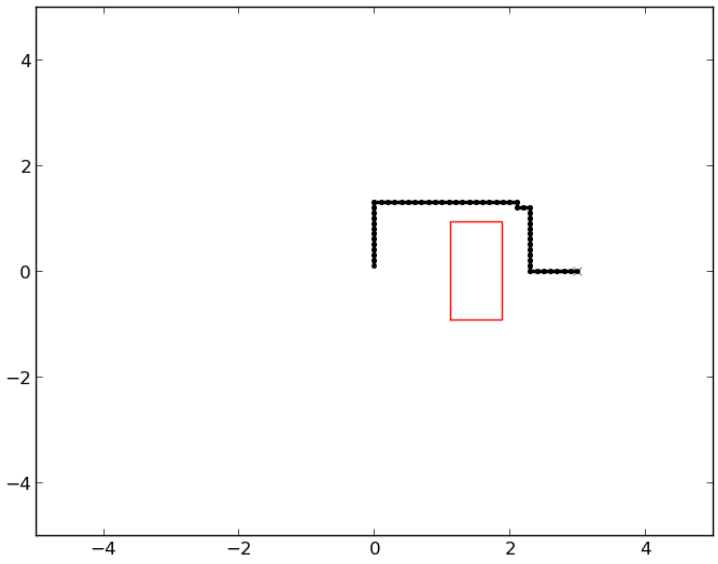

Figure 1: Example path found by the 2D breadth-first search algorithm using a resolution of 0.1.

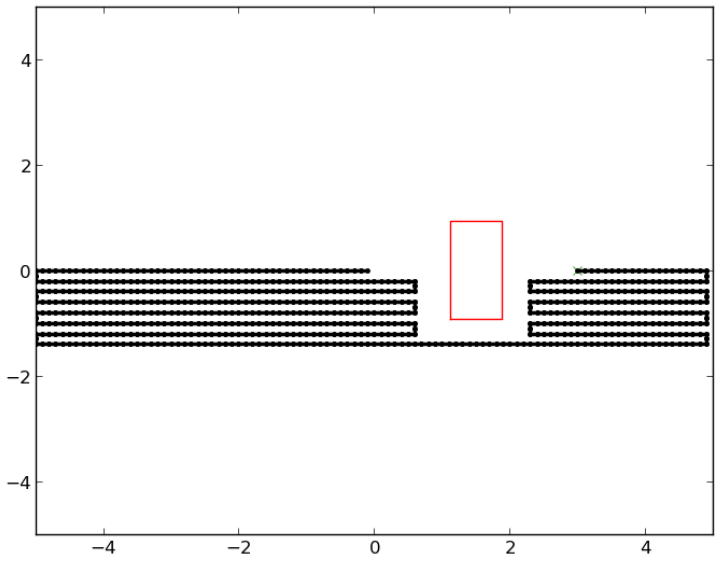

Figure 2: Example path found by the 2D depth-first search algorithm using a resolution of 0.1.

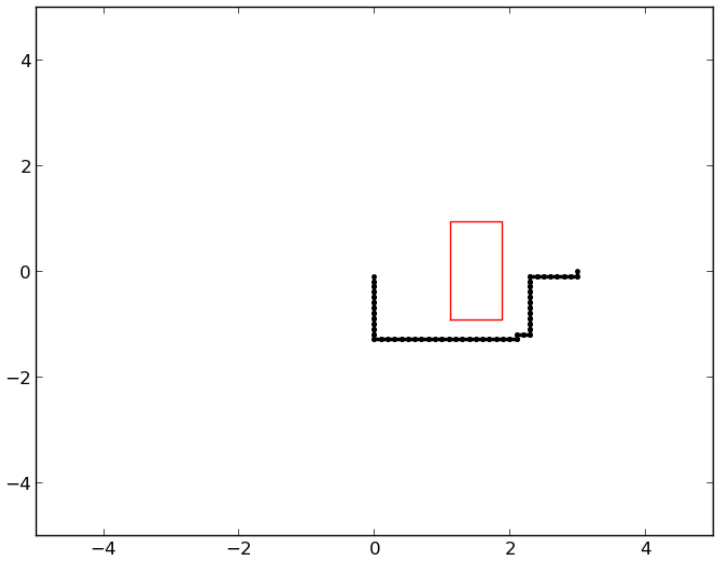

Figure 3: Example path found by the 2D A-Star search algorithm using a resolution of 0.1.

Q2

We ran our A-Star search algorithm on HERB's 7 DOF WAM arm at a resolution of 0.1. The results can be found in Table 4.

Refer to the file *7d_astar.ogv* for an example video of one of our A-Star search algorithm runs on the WAM arm.

Table 4: 7D WAM Arm A-Star Search Results

Parameter	Res = 0.1
Path Length	1.700 rad
Plan Time	17.001 s
Nodes in Path	18
Closed List Size	114

Q3

In the previous homework, we implemented an RRT algorithm with path shortening that found a collision-free path for our robot to follow in the 2D environment. This RRT code is not included in our submission. Please see Vamshi Korundi's HW2 submission for reference. Using this algorithm with the same start and goal positions in this homework, we ran the RRT three times. The results can be found in Table 5.

We see that for the 2D case A-Star consistently returns shorter paths than RRT. This makes sense, as A-Star finds the optimal path given your discretization, while RRT simply finds a path. Path shortening improves the RRT's path, but it is still fundamentally random and inefficient in terms of navigating from start to goal.

A-Star is also thought of as a fast algorithm, but in this 2D environment case we see that RRT is faster on average, even compared to the coarse 0.25 resolution A-Star case. The RRT used is aggressive. It's paths are long, but contain very few nodes and very few nodes are searched before finding a solution.

Using coarser resolutions on A-Star sees it's plan time approach that of RRT's, but the space is so open that the RRT generally checks less than 10 nodes before finding a path. Of course, it is inherently random and is not guaranteed to be faster than A-Star all the time. Also, in a space with more obstacles RRT would have to sample many more times to find collision free paths, while A-Star would search the space more efficiently.

Table 5: 2D RRT Planner with Path Shortening Results

Parameter	Trial 1	Trial 2	Trial 3	Average
Path Length	11.419 m	9.025 m	8.438 m	9.627 m
Plan Time	0.086 s	0.054 s	0.042 s	0.061 s
Number of Nodes	4	4	4	4

We did the same with the 7D WAM Arm environment. We found that our RRT algorithms did not finish in a reasonable time with this 7D goal, but the RRT connect algorithms we wrote earlier do. The concept of randomly searching the space is maintained. This RRT-Connect code is not included in our submission. Please see Sun Chunyang's HW2 submission for reference. The results can be found in Table 6.

We see a similar pattern in 7D as we did in 2D. Yes, A-Star returns a shorter, more optimal path. But it takes longer to plan in this case because the obstacle space is not dense. The discretization is fairly fine and requires a number of nodes be checked along the way to the goal. The RRT is coarse and aggressive and only needs to check a few random nodes before finding a set that connects the start to the goal.

Again, RRT is inherintly random and not guaranteed to be faster than A-Star, but in 2D and higher dimensional spaces with sparse obstacles, RRT can find a path quite quickly.

Table 6: 7D WAM Arm RRT Planner with Path Shortening Results

Parameter	Trial 1	Trial 2	Trial 3	Average
Path Length	6.296 rad	7.349 rad	6.019 rad	6.555 rad
Plan Time	0.056 s	0.075 s	0.196 s	0.109 s
Number of Nodes	3	3	4	3.333

Q4

We have heuristic RRT working on 2D environment. We didn't record the result in 7D as it took a long time to run. From the 2D performance, we see that heuristic RRT does not give an optimal path as Astar did. It is also not as fast as Vamshi's RRT, but that is because Vamshi's RRT was highly optimized. Compared to a similar RRT (like Steven's hw2), it is faster and returns shorter paths. Below table shows the results 7. Figure 4 illustrates a 2D plan with hrrt.

Table 7: 2D heuristic RRT with Path Shortening Results

Parameter	Trial 1	Trial 2	Trial 3	Average
Path Length	8.017 m	10.242 m	8.471 m	8.91 m
Plan Time	1.925 s	1.300 s	1.539 s	1.588 s
Number of Nodes	4	3	4	3.667

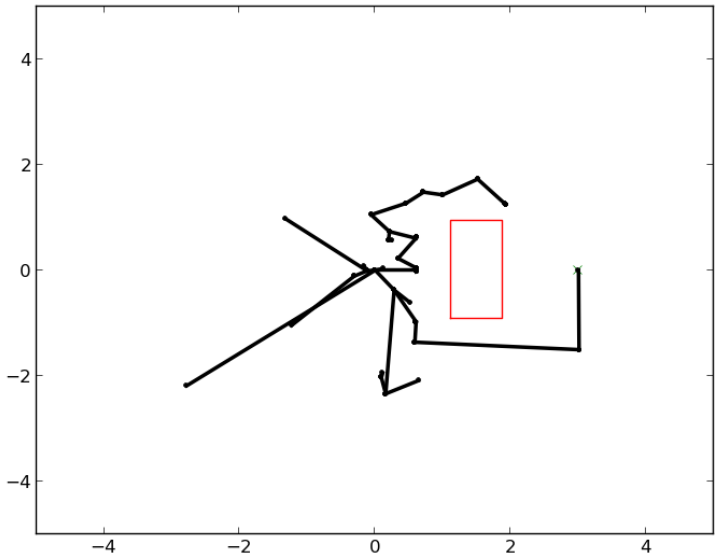

Figure 4: Example path found by the hrrt planner.