Caelan Garrett
6.S078 Assignment 2
09/26/13
Using 1st Late Day

Description

I used the star algorithm to compute the c-space convex obstacles. I computed these c-space obstacles for each theta orientation slice in the user specific list of slices (which includes the start and end orientations). Storing these transformations in a table allowed me to quickly retrieve the necessary slice without re-computing them.

For each slice of the c-space, I first compute the visibility graph in that c-space assuming no rotations in the slice. I do this by the standard approach of testing line intersections between vertices of the c-space obstacles with the c-space obstacles themselves. I ignore lines that connect vertices of the same c-space polygon unless they are edges. I also avoid testing the same line twice in one slice (couldn't avoid the rhyme).

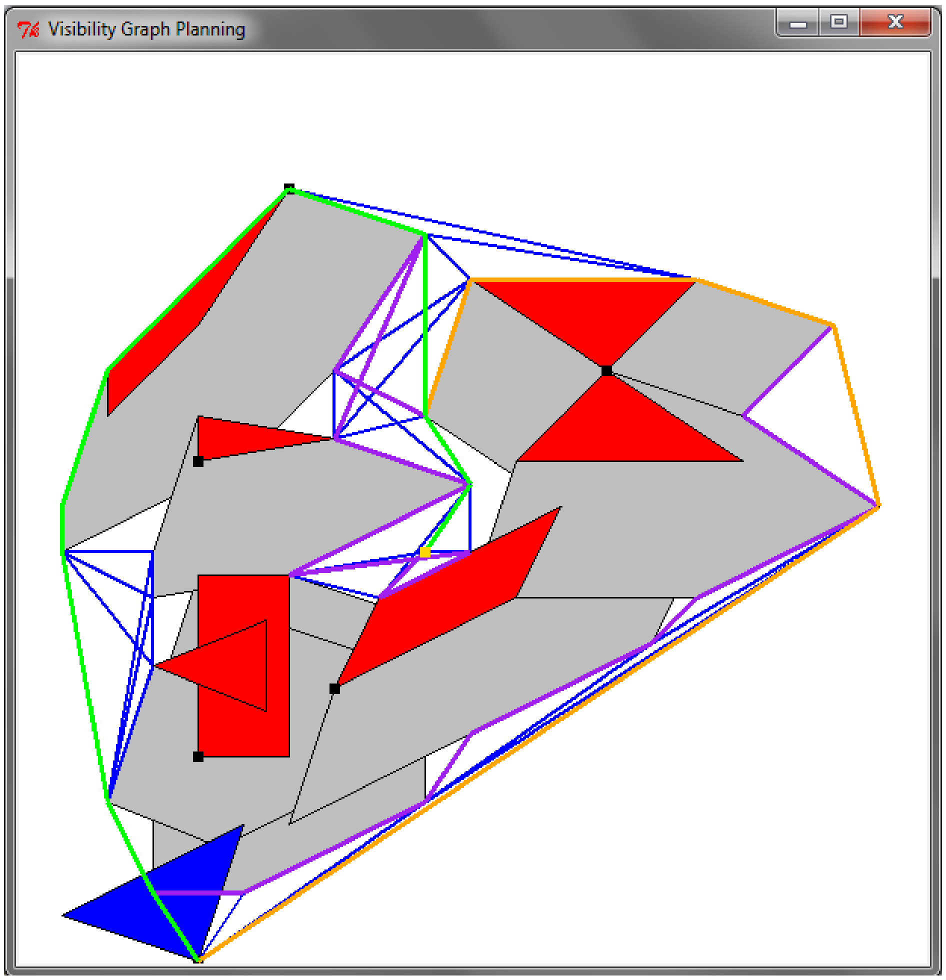
To connect these slices, I add edges in the total visibility graph between slices if rotation around the robot's reference point to the above and below slice is collision free (this is only tested by ensuring the start and end configurations are collision free to simplify the operation). If the vertices do not align in each c-space vertically, I add each vertex to the above and below visibility graph slices and add corresponding edges within that slice to the vertex.

I used the upper-bound on the farthest distance any point moved between configurations as my metric. For translations, this was just the Euclidean distance. For rotations, this was the product of the difference in angle with the radius of the robot (the farthest distance on the robot from the reference). This distance metric was also my admissible heuristic. 10 times the distance metric was my not admissible heuristic.

Unlike the previous example, BFS and the not admissible A* frequently didn't find the shortest path. Because BFS is no longer on an equally spaced grid, the number edges did not scale with the distance so it has no guarantees on shortest paths. Rotations seemed to fool the not admissible A* into greedily moving to what turned out to be loner paths.

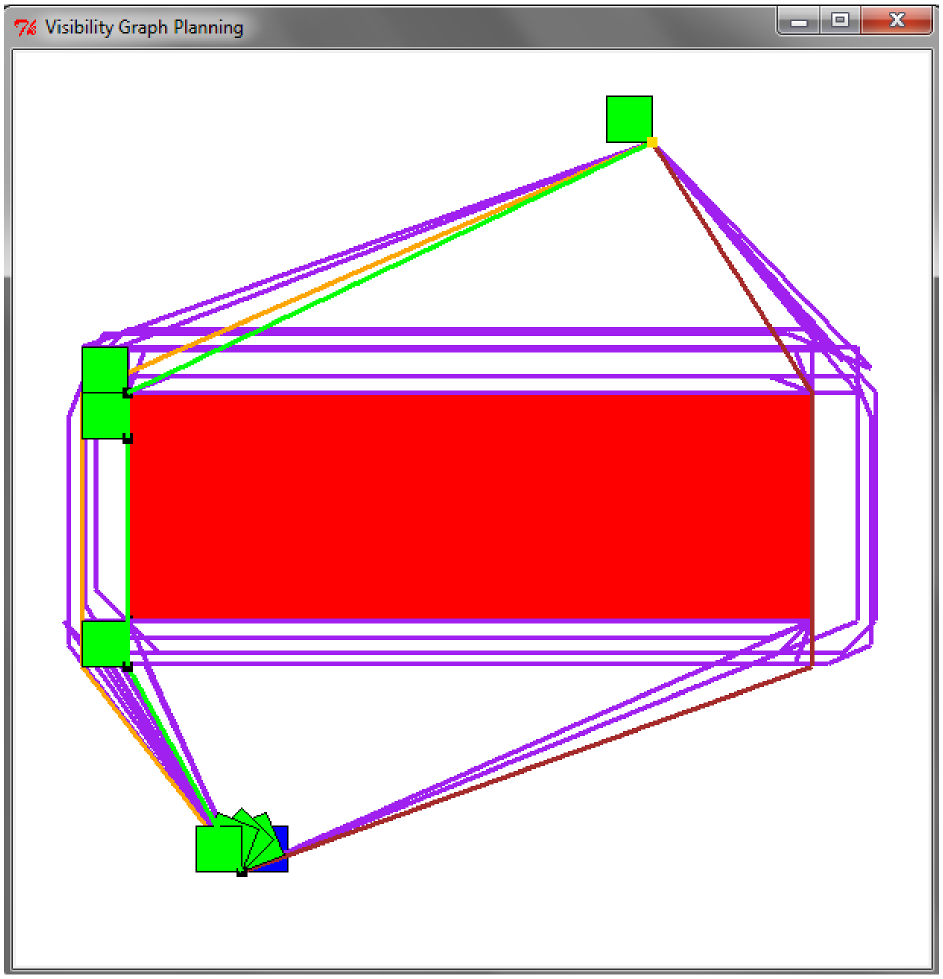
Geometrically, I added Axis Aligned Bounding Boxes to geometric primitives to speed up computations (line/line collisions, line/polygon collisions, point/polygon containment, etc…). I colored in the obstacles and the robot to make it clearer. The robot is blue, and obstacles are red. Reference points are drawn on the Objects. The goal is gold. A path determined by DFS is shown in purple. A path determined by BS is shown in orange. An optimal path generated by UCS (or A*) is shown in green. Finally, a path generated by A* with a not admissible heuristic is drawn in brown.

2D Visibility Graph in Hard Difficultly World – Here, I am only planning using translations in the Cartesian plane to better visualize the c-space objects and visibility graph. Grey polygons are the c-space obstacles, and blue lines represent the visibility graph. The visibility graph took 1.005 seconds to generate.



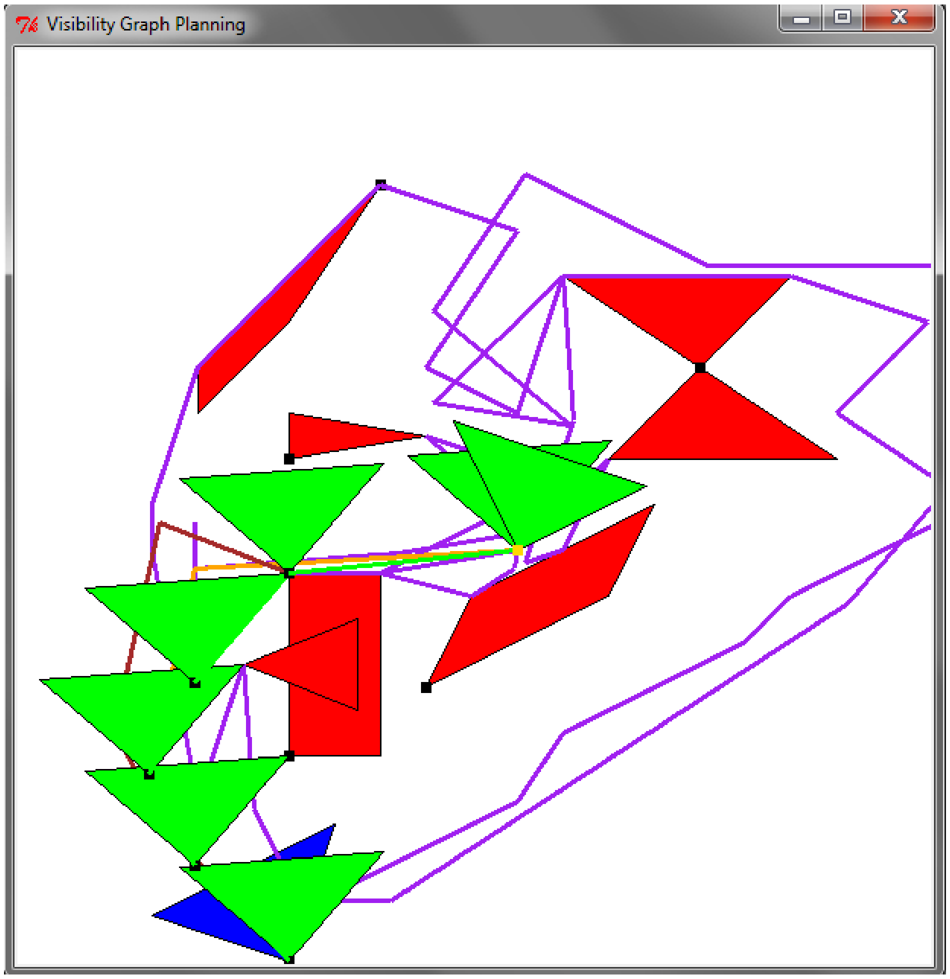
Search Algorithm	Run time (s)	Path length
DFS	0.002	64.276
BFS	0.002	37.080
UCS	0.004	30.215
A* with Not Admissible Heuristic	0.003	30.215
A* with Admissible Heuristic	0.003	30.215

3D Visibility Graph in Medium Difficulty World – This planning occurred in the standard 3 dimensional c-space. The orientation is divided into 16 slices. The c-space obstacles and visibility graph are not drawn to remove clutter. Instead, a trace of the robot following the optimal path (in green) is shown. The visibility graph took 0.137 seconds to generate.



Search Algorithm	Run time (s)	Path length
DFS	0.015	570.364
BFS	0.014	27.578
UCS	0.039	26.116
A* with Not Admissible Heuristic	0.002	28.025
A* with Admissible Heuristic	0.007	26.116

3D Visibility Graph in Hard Difficulty World – This planning occurred in the standard 3 dimensional c-space. The orientation is divided into 16 slices. The c-space obstacles and visibility graph are not drawn to remove clutter. Instead, a trace of the robot following the optimal path (in green) is shown. The visibility graph took 10.461 seconds to generate.



Search Algorithm	Run time (s)	Path length
DFS	0.004	169.656
BFS	0.014	19.370
UCS	0.033	18.100
A* with Not Admissible Heuristic	0.006	24.053
A* with Admissible Heuristic	0.008	18.100