COMS W4733: Computational Aspects of Robotics

Homework 4

Due: April 8, 2019

This assignment consists of two parts. Part 1 should be completed and submitted individually. Part 2 may be completed either by yourself or with one other student. If working with a partner, please be sure to use the "Group submission" option on Gradescope.

1 Reading on Decomposition and Potential Fields (50 points)

Carefully read "Mobile Robot Path Planning Based on Hierarchical Hexagonal Decomposition and Artificial Potential Fields" by E. Hou and D. Zheng (Journal of Robotic Systems, 1994). Please provide responses to the following. Note that there may not necessarily be a single correct answer—you should be able to provide thoughtful and critical responses from your reading of the paper.

1. What are the difficulties enumerated by the authors for C-space graph search methods? Does the technique described in this paper address or potentially alleviate these difficulties in any way? Does hexagonal decomposition present similar drawbacks?

2. At the end of Section 3.1, the authors note the possibility that a path may not exist within the current region of influence, which can be resolved by enlarging the region. Provide a description of how you would efficiently implement this procedure and combine it with the overall search algorithm.

3. At the end of Section 4.1, the authors describe two tests to determine whether a hexagon is IMPASSABLE. The second test checks whether an obstacle vertex lies within a set of dashed lines in Figure 8b. Provide a description of how you would actually implement this test.

4. The authors address the issue of local minima of the potential fields in Section 4.4. A strategy to avoid multi-point cycles is to mark already-visited hexagons as IMPASSABLE, even if they are actually PASSABLE. Can there be situations in which this would lead to an incorrect outcome or a worse overall path? If not, argue why this is correct.

5. Consider the simulation result shown in Figure 12. Why is the robot initially attracted toward the dead end on the left-hand side? Assuming you have knowledge of the environment and obstacles apriori, what would you change about the setup of the algorithm to avoid the local minimum in the first place? Be specific, *e.g.*, particular components or values of the potential functions at various obstacles or locations.

2 Implementing VGRAPH (100 points)

You will be implementing the visibility graph path planning algorithm using the ArbotiX Turtlebot simulator. The resulting graph and robot path following will then be visualized using RViz.

2.1 Data Files

All units in the data files are cm. RViz grid cells are 50cm by 50cm. The map itself is fixed to be 1200cm by 600cm, with top left being $(-300, 300)$ and bottom right being $(900, -300)$. The robot starts at $(0, 0)$ facing the positive x direction. The goal position is defined in `data/goal.txt`.

Obstacles are defined in `data/world_obstacles.txt` and shown as polygons on the map. The first line of the file indicates the number of obstacles. Each subsequent line corresponds to an obstacle, with the first number specifying number of vertices, and remaining numbers specifying vertex locations in counterclockwise order.

2.2 Setting Up

To get started, download the provided files, which will provide a custom ROS package that can be added to your workspace. Then run the following commands inside the directory:

```
catkin_make
source devel/setup.bash
rospack profile
```

The files `data/world_obstacles.txt` and `data/goal.txt` contain the world's obstacles and goal position. You can modify these to test your code in different settings. If you decide to do so, run the following commands to generate a new `map.png` file:

```
roscd vgraph/src
python create_map.py
```

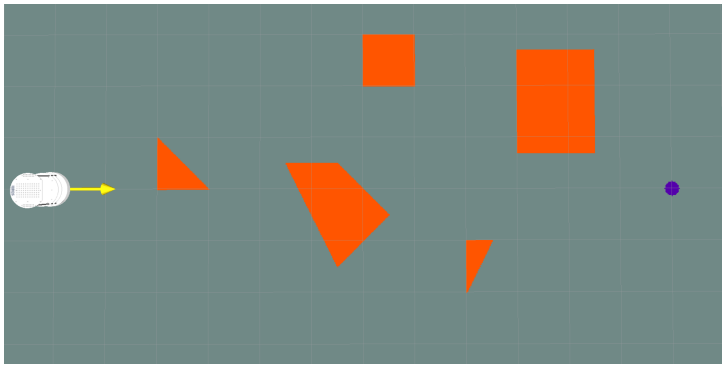
Finally, you can launch ROS by running

```
roslaunch vgraph launch.launch
```

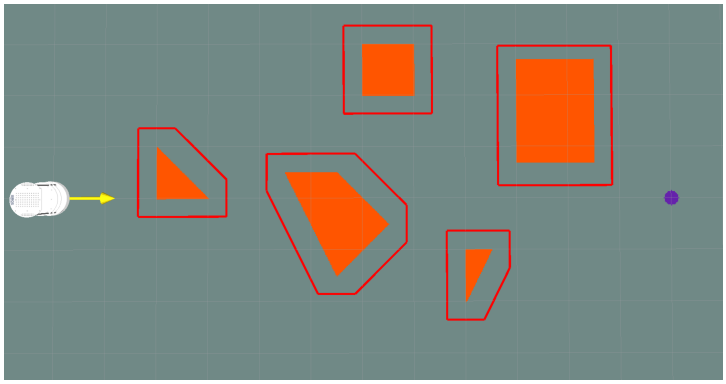
Remember that you'll want a new terminal to run further commands.

2.3 Tasks

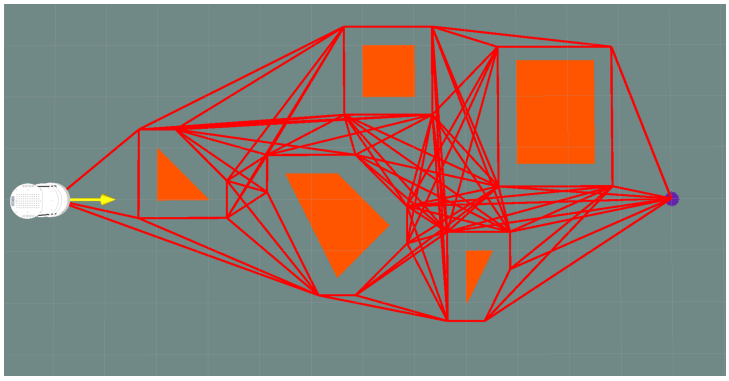
Place any Python files that you create in `/src`. Visualizations can be done using visualization_msgs/Marker or visualization_msgs/MarkerArray messages. When you launch the simulation using the provided data files, you should see the following environment.



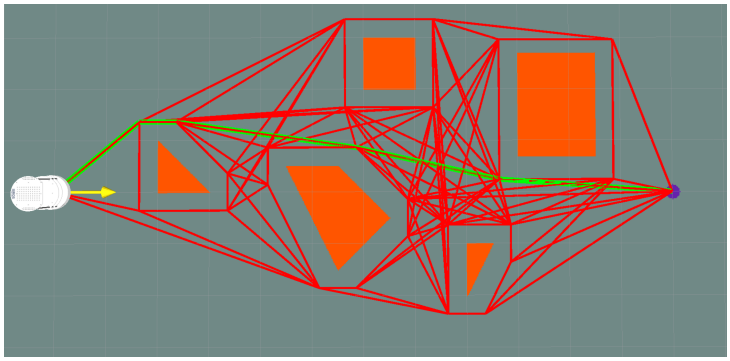
1. Grow the obstacles using a reflection algorithm, assuming the robot to be a 36cm by 36cm square. You can choose the robot reference point anywhere in this square. As a hint, you may want to check out the ConvexHull package in SciPy. The following example picks the square center as a reference point.



2. Create the visibility graph by a) connecting all obstacle vertices and the start and the goal locations and then b) using a collision checker to remove edges that collide with obstacles (excluding endpoints). You may use any existing libraries or packages for this task. The generated VGRAPH for the example above should look as follows.



3. Implement or use a shortest path finding algorithm to find the shortest path from start to goal. Again, you may use any existing libraries or packages for this task.



4. Implement a simple motion controller to have the robot move along the path you followed. One option is to refer to odom_out_and_back.py.

2.4 Submission

Zip up your `vgraph` package with all files needed to run your code, including those that you did not modify. Please include the following two additions:

- A video (or a link to a video) of the robot successfully following the shortest path in the VGRAPH.
- A writeup explaining your implementation, including any interesting observations and choice of packages. Also be sure to explain how to run your code, including functionalities of new files that you created.