Lab 5: Path Planning

CSCI 3302: Introduction to Robotics

Report due 03/17/20 @ 23:59pm

Total points: 100 + extra credit for early submission

The goals of this lab are:

- Implement a shortest-path finding algorithm (Dijkstra's algorithm)
- Use shortest path information from Dijkstra to generate an actual route to follow

You need

- Lab 5 Base Code (lab5_base.py)
- obstacles_test1.png and obstacles_test2.png

Dijkstra's algorithm takes a graph as input and calculates the shortest path from every single vertex back to a user-specified source vertex. See https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm#Pseudocode for example code. For path planning purposes, treat every cell of your 2D map as a node of a graph with edge costs that are inferred from the map geometry and presence of obstacles. Provided with a graph, source vertex, and a destination vertex, your code will find the shortest collision-free path from the source to the destination.

Part 1

1. Implement a version of Dijkstra's algorithm into the base code. In particular, use your group's cost function from Lab 4 to compute the 'distance' between two vertices.
   [Hint: Test using a simple 4x4 map with some hardcoded obstacles to validate that Dijkstra's algorithm returns the correct distances!]

2. Write code that returns a sequence of vertices leading from a source to destination vertex.
   [Hint: Make sure your solution gracefully handles the case where there is no valid path between source and destination vertices!]

Part 2

1. This time you will develop additional code to run Dijkstra's on a real-world image with obstacles. You will have to convert the matrix that represents the image pixel intensities to a simpler grid world so that you can apply the earlier developed Dijkstra's on that. You will have to come up with a strategy to decide whether a cell is an 'Obstacle' cell or a 'Free' cell.
   'pixel_grid' has the intensities such that darker pixels have a higher value.

You can run part 2 by running

python lab5_base.py -o obstacles_test1.png -s 1.2 0.2 -g 0.3 0.7

where -o denotes the obstacle file -s denotes the source and -g denotes the goal/destination

The source and goal are given in meters. Assume the lower left corner of the image as the origin for the world coordinate and the +ve x-axis is from lower left to lower right and the +ve y-axis is from lower left to upper left direction of the image.
The images are 1.8 meter x 1.2 meters.

2. Develop the code to visualize the planned path on the image. You can either show the path using superimposed lines or color the path on your grid and superimpose the grid with the identified path. Feel free to come up with your own visualization!!

Part 3: Lab Report

1. Briefly explain how the algorithm works (N.B. This is a very common interview question).
2. What is an example of an admissible heuristic that would help with implementing an informed search algorithm for this problem?
3. [Part 1] Show sample output from Q1 (a 4x4 cost matrix given a start and destination state)
4. [Part 1] Show sample output for the example used in Q2.
5. [Part 2] Show the paths on the images for the following 4 cases.

   a. **obstacles_test1.png**, source = **(1.2, 0.2)**, goal = **(0.225, 0.975)**
   b. **obstacles_test1.png**, source = **(0.9, 0.3)**, goal = **(0.9, 0.75)**
   c. **obstacles_test2.png**, source = **(1.2, 0.2)**, goal = **(0.225, 0.975)**
   d. **obstacles_test2.png**, source = **(0.225, 0.6)**, goal = **(1.35, 0.3)**

6. Roughly how much time did you spend programming this lab?