Lab 6: Path Planning II

CSCI 3302: Introduction to Robotics

Report due 3/31/20 @ 23:59pm

(Early turn-in bonus: +2pts extra credit per day)

The goals of this lab are:

- Use shortest path information from Dijkstra to generate a discrete path
- Turn a discrete path into a list of waypoints a robot can follow
- Showcase higher-order planning by having Sparki sim plan and execute a collision-free path

You need:

- sparki-simulator-v2.py (updated one, please download the latest one)
- Your solutions for **Lab 3 (for IK)**, **Lab 4 (for working with sparki sim)** and **Lab 5 (Path Planning)**.

Overview

Dijkstra's algorithm takes a graph with N nodes as an input and returns the cost of reaching every single vertex from a user-specified source vertex. Using this information, we can find a path of vertices connecting a start location to a goal location. In order to make use of this information, we have to convert the vertices on this path into coordinates (**goal pose for Lab 3**) we can travel to. Recall that with inverse kinematics we can turn a desired goal pose into the wheel rotations that will enable the robot to travel there! Therefore, combining these two ideas will allow us to use path planning on a graph to enable a robot to traverse a physical (virtual in this case), continuous space.

Instructions

Each group must develop their own software implementation and turn in a **single report** that contains:

- The code (1 .py file **which would be an extension on your Lab 5 solution**)
- One individual lab report in PDF format. Please put the number of each question next to your answers, rather than turning in your answers as an essay
- A video of your code running on the Sparki Sim (1 file). If your video file is bigger than 50mb, please submit a link to google drive or some other cloud service of your choice.

If your group does not finish the implementation by the end of the class, you may continue this lab on your own time as a complete group.

Part 1: Path Planning

1. Convert your Dijkstra's output into waypoints (intermediate goals) with respect to the world coordinate frame (refer to Lab 5).

Part 2: Path Following

1. Start your simulator by typing
   python sparki-simulator-v2.py -o obstacles_test1.png -w empty_world.png
   You can change the obstacles by changing the file name. Keep the world file constant as you don't want a circuit in this case.

2. Code up the loop() function similar to the one you implemented in Lab 3 (IK lab). You may also want to review your Lab 4 (the Sparki sim mapping lab) for python implementation of the loop.

3. Set up the initial pose to the input source.
   (Portion of the last lab that helps with the input and coordinate system)
   ***python lab5_base.py -o obstacles_test1.png -s* 0.9 0.3 *-g* 0.9 0.75**
   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.

4. Use either of the controllers that you built in Lab 3 (Part 2 or part 3). Note that you have to reach your final goal by following intermediate waypoints. You don't necessarily have to fix the heading error as you don't care about the goal orientation in this case. However, if your target theta points to the next cell already, it will be easier for you to get to the next cell!!
   (You can use a state machine within your loop for managing the code. We encourage this to make your single page code more modular.)

Part 3: Lab Report

1. How could you modify this code to gracefully handle unforeseen objects in Sparki's path?
2. Print the output of the waypoints in terms of world frame for the following 4 scenarios. They are the same ones from the previous lab.

1. **obstacles_test1.png**, source = **(1.2, 0.2)**, goal = **(0.225, 0.975)**
2. **obstacles_test1.png**, source = **(0.9, 0.3)**, goal = **(0.9, 0.75)**
3. **obstacles_test2.png**, source = **(1.2, 0.2)**, goal = **(0.225, 0.975)**
4. **obstacles_test2.png**, source = **(0.225, 0.6)**, goal = **(1.35, 0.3)**

3. Run your code for the **1.** and **4.** case from previous question and submit the videos for both.
4. What are the names of everyone in your lab group?
5. Roughly how much time did you spend programming this lab?

To be (optionally) submitted separately via e-mail to Prof. Roncone: aroncone@colorado.edu

(Optional, confidential, not for credit) A brief description of any problems (either technical or collaborative) encountered while performing this lab (e.g., issues with the clarity of instructions, clarity of documentation, lab colleague's behavior, etc.)