Lab 3: Trajectory Planning

MEAM 520, University of Pennsylvania

October 3, 2017

This exercise is due on **Wednesday, October 18, by midnight (11:59 p.m.)** Late submissions will be accepted until midnight on Friday, October 20, but they will be penalized by 10% for each partial or full day late. After the late deadline, no further assignments may be submitted; post a private message on Piazza to request an extension if you need one due to a special situation such as illness. This assignment is worth 25 points.

You may talk with other students about this assignment, ask the teaching team questions, use a calculator and other tools, and consult outside sources such as the Internet. To help you actually learn the material, what you submit must be your own work, not copied from any other individual or team. Any submissions suspected of violating Penn's Code of Academic Integrity will be reported to the Office of Student Conduct. When you get stuck, post a question on Piazza or go to office hours!

Individual vs. Pair Programming

You may do this assignment either individually or with a partner. If you do this lab with a partner, you may work with anyone you choose, but you must work with them for all parts of this assignment. Looking for a partner? Try the 'Search for Teammates!' tool on Piazza.

If you are in a pair, you will both turn in the same report and code (see Submission Instructions below), for which you are jointly responsible and you will both receive the same grade. Work closely with your partner throughout the lab, following these guidelines, which were adapted from "All I really needed to know about pair programming I learned in kindergarten," by Wililams and Kessler, *Communications of the ACM*, May 2000. This article is available on Canvas under Files / Supplemental Material.

- Start with a good attitude, setting aside any skepticism, and expect to jell with your partner.
- Don't start alone. Arrange a meeting with your partner as soon as you can.
- Use just one setup, and sit side by side. For a programming component, a desktop computer with a large monitor is better than a laptop. Make sure both partners can see the screen.
- At each instant, one partner should be driving (writing, using the mouse/keyboard, moving the robot) while the other is continuously reviewing the work (thinking and making suggestions).
- Change driving/reviewing roles at least every 30 minutes, *even if one partner is much more experienced than the other.* You may want to set a timer to help you remember to switch.
- If you notice an error in the equation or code that your partner is writing, wait until they finish the line to correct them.
- Stay focused and on-task the whole time you are working together.
- Take a break periodically to refresh your perspective.
- Share responsibility for your project; avoid blaming either partner for challenges you run into.
- Recognize that working in pairs usually takes more time than working alone, but it produces better work, deeper learning, and a more positive experience for the participants.

Planning Trajectories for the Lynx Robot

This lab is about planning collision-free trajectories through a cluttered environment using the Lynx robot.

Method

Consider the Lynx robot trying to move its end effector from a point p_{start} in its workspace to another point p_{end}. The environment has i spherical obstacles in it, which are centered at positions $[x_i, y_i, z_i]^\intercal$, have radius r_i, and are attached to the boundaries of the environment using straight wires (line segments). Devise a planner that will allow the robot to perform this movement without colliding with any of the obstacles. Questions to consider include:

- How to represent the workspace
- How to check for collisions (with the end effector? with the rest of the robot?)
- How to account for the geometry (volume) of the robot
- How to compute waypoints
- How to interpolate between waypoints

Explain your choices. Include your pseudocode in the report.

Evaluation

Design a few tests (environments, start and end positions) to check whether your algorithms work. Explain why you have chosen those tests. Run each test several times to evaluate the performance of your planner. Consider questions such as:

- How often does your path planner succeed?
- How long does it take for your planner to find a path (running time)? MATLAB's in-built functions `tic` and `toc` may be useful here.
- When your planner finds a path, is the path the same over multiple runs?
- How do the answers to these questions change for diffferent environments or variations in your implementation?

You may reuse the Lynx starter code from Lab 2, as well as any functions you have written for previous labs. We suggest that you create a new function called `lynxplanner` that takes a start position, an end position, and a list of obstacles as input and generates a plan for the robot to execute. Correct IK code from Lab 2 will be posted after the Lab 2 late deadline on Oct. 6.

For the evaluation, you can use simulation and/or physical hardware. Be sure to justify why you chose one or the other or both. For the physical hardware, we have set up some obstacle courses that you can use (expected Oct. 9).

- Station 1: our sample obstacle course; do not move obstacles
- Station 2: no obstacle course; you can use this station to test your algorithms without worrying about physical collisions before you test on another station
- Station 3: reconfigurable obstacle course; feel free to rearrange or change out the obstacles to test your algorithm on a custom environment

Analysis

Discuss any conclusions on your planner based on the evaluations. What kinds of environments/situations did your planner work well for? What kinds of environments/situations was it bad at? What issues did you run into or what differences were there between your expected and experimental performance? What changes would you make to your implementation if you had more time with the lab?

Submission Instructions

Register a lab group. If you worked on the lab in a pair, follow these instructions to make sure that both members receive credit. If you did the lab by yourself, then don't worry these instructions; you just need to submit the files. Since we do not enforce that you keep the same lab group for the entire semester, you will need to do this for every lab assignment. In order to ensure that Canvas recognizes your group, register your lab group **before** submitting the assignment.

1. On Canvas, click on the link 'People' in the left navigation bar.
2. Click on the 'Lab 3-' tab at the top of the page.
3. Click and drag the names of both team members into an empty lab group.
4. When one member submits the assignment to Canvas, both members should check that they have access to the files.

Clean your code. Your code should be cleaned up so that it is easy to follow. Remove any commented-out commands that you are not using, and add comments to explain the tricky steps. Clearly indicate which parts of the code correspond to which parts of the lab.

Submit the assignment. You should submit a **pdf report** on Canvas containing your solutions and analysis for these tasks, as well as a **zip file containing all of your code**. These should be 2 separate files. **Do not include your pdf in the zip.** The format of the report is up to you, but you should make sure that it is clear, organized, and readable.