Lab 2 :Path Following Controller for Mobile Robots

MEAM 520, University of Pennsylvania

June 25, 2020

This exercise is due on **Monday, July 6 by midnight (11:59 p.m.)** Submit your answers to the questions in this document as a pdf on Gradescope. Late submissions will be accepted until midnight on **Thursday, July 9**, but they will be penalized by 25% 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 30 points.

You may do the exercise in pairs and add your partner on Gradescope to ensure both students receive credit. 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. When you get stuck, post a question on Piazza or go to office hours!

1 Concepts

This lab relies on an understanding of the kinematics and control of differential drive mobile robots. Please read through the handout `MobileRobots.pdf` which can be found on Canvas. This document will be a valuable reference for this lab as it contains an in depth discussion of these concepts.

2 Coding Assignment

1. **Setup**: In the virtual machine, download the `lab2.zip` starter code for this assignment from Canvas. Extract this folder to a new directory. This zipped file contains the following files:

   - `mobile_robot.py`: this file abstracts away all communication with ROS/Gazebo and it is not intended to be modified.
   - `mobile_sandbox.py`: this file is intended to act as the "main" function for this project and contains the code to initialize the simulation and run the control loop for the mobile robot. It also contains prototype code to plot the output from the simulations. This file should be modified as necessary.
   - `mobile_controller.py`: this file contains the skeleton for the Controller class and will need to be completed to control the mobile robot.
   - `mobile_path.py`: this file contains the skeleton for the Path class and will need to be completed to control the mobile robot.

   After inspecting the files in this folder proceed to the following steps and complete the associated programming tasks, using the documentation provided in the code as a guideline. Note that although a class based architecture is suggested for completing this assignment you are welcome to implement a solution for this lab in any way that you see fit. If you deviate from the suggested implementation however, be sure to document your code extra well so that it is easy to follow when grading.

2. **Path Following Controller**: The first step towards developing a path following mobile robot will be defining the algorithm to control it. Open the file `mobile_controller.py` in a text editor and complete

the implementation for the controller class using the controller outlined in `MobileRobots.pdf`. This will require the writing of two functions

- `__init__`: this function is called when a Controller object is created, and should set the intrinsic parameters of the controller
- `update`: a function to map the state and state derivative of the robot to its wheel velocities

3. **Path Definition**: Next, define the path for the controller to take. Complete the implementation for the Path class inside the file `mobile_path.py`. The path class should compute the estimated track error and the Jacobians and save as its attributes. We are interested in testing several different paths for the robot to travel along, but until you are confident that your controller is implemented correctly it is recommended that the paths you follow are simple (such as the line $y = 0$). You will return later in this lab to edit this class several times to support following other paths.

3 Simulation

In order to test this controller in simulation, we will need to open a new Gazebo environment. Open a new terminal (press Ctrl+Alt+T, or right click, find "Open Terminal") and run `meam520_update` to update the system, this will download some files and packages for Lab 2 automatically. Then run `meam520_mobile` in the terminal to start the Gazebo simulation.

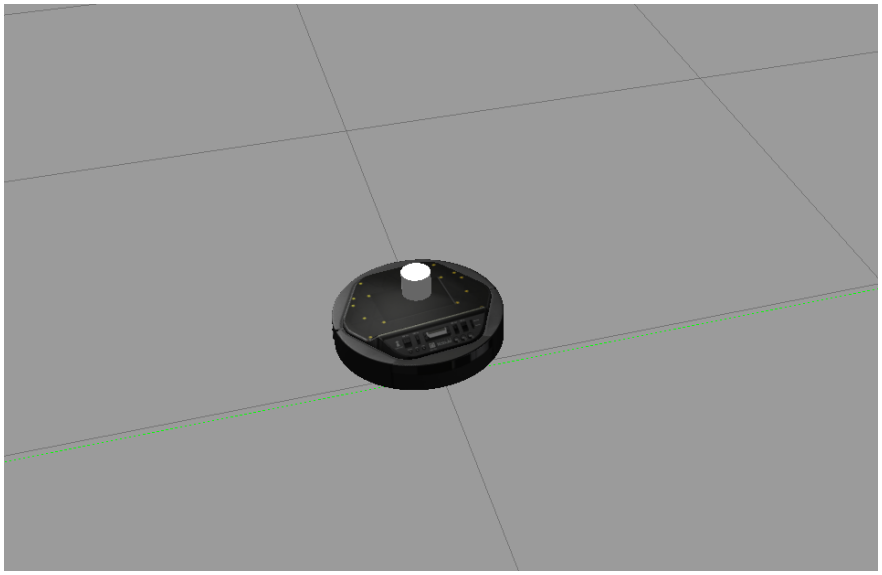

Figure 1: The mobile robot in the Gazebo simulation environment.

To run the code you just wrote open the file `mobile_sandbox` in Spyder3 and press the play button in the top ribbon (or use shortcut F5) to run the file. This file should output some messages updating you on its status. The program is running as intended if the output you receive is similar to the following messages.

```
Starting ROS thread...
ROS thread started
Waiting for Gazebo...
Waiting for Gazebo...
Setup complete - entering loop
```

You should be able to open up the Gazebo window and watch the mobile robot moving. If you implemented the Controller and Path classes correctly the robot should move along the specified path. When you want to end the simulation you can go back into Spyder3, click on the iPython console, then press Control+C. Upon doing this, an output message will read `Exiting` and the robot will stop moving in Gazebo.

3.1 Experiments to Conduct in Simulation

Test different gains for your controller and determine a set of parameters that work well.

1. Go back into `mobile_path.py` and modify the class so that the robot can follow the paths described below.

   (a) The line $y = 2x + 1$

   (b) A circle of radius 1 centered about the origin

   (c) The polynomial $y = x^3 - x^2 - x + 1$

   (d) Spinning the robot in place

   For each experiment initialize the robot to the point $(0, 1)$. Plot the path your robot follows by plotting its position in y vs. x. Include the code used to generate the Path with this plot.

2. Test different initial conditions for the robot and plot the error of your robot over a 30 second time period. Include the following cases in your experiments.

   (a) On the desired path

   (b) Near the desired path

   (c) Far from the desired path

   (d) In the middle of a circle

3. Test different desired speeds for the robot to travel at. Compare the error over time caused by using different values for this parameter.

4. Test different lengths for the variable l, the offset distance from the axle of the point that is being tracked. Compare the error over time caused by using different values for this parameter.

3.2 Questions to Think About

Based on the results of your simulation, think about the following questions.

1. What are some advantages and disadvantages of using this controller for a mobile robot? How do you measure success of a particular controller? Come up with a few qualities you would like to see in a successful controller and then ways to quantify these attributes.

2. What strategy did you employ to choose values for the k_p and k_i parameters, and how did they affect the controller performance?

3. For what conditions does this controller succeed in following the path? Which conditions does the controller not succeed? These conditions might include restrictions on initial state, desired path or speed, the variable l, etc.

4. What are some differences you might expect to encounter between deploying this controller in simulation versus hardware?

5. How might you use this controller to travel through some path of given waypoints? Outline a strategy for tackling this problem.

4 Submission Instructions

You should submit a **pdf report** on Gradescope as well as a **zip file containing your code**. These should be 2 separate files. **Do not include your pdf in the zip.**

4.1 Report

The report should consist of:

1. A short explanation of the code you have written. Include pointers to which major functions in your code perform what work. This will help the graders understand and provide feedback on your work. (This description can be bulleted. No need to use full sentences.)
2. Documentation of the experiments that were conducted, including all the information necessary to replicate your experiments and the relevant output from the experiments (plots, data, etc).
3. Your analysis of your simulation results and a discussion prompted by the Questions to Think About.

The format of the report is up to you, but you should make sure that it is clear, organized, and readable.

4.2 Code Submission

Your code submission should include **all** the python files required to run the simulation, zipped in a single folder.