ENGN4627 Lab 2: Robot Control

1 Overview

In this lab, you will complete the following tasks:

- Write a procedure to calibrate your robot's parameters (scale parameter and wheel track)
- Develop an algorithm your robot can use to follow a line by processing the robot's camera output.

For this lab, you need to submit your report on wattle.

2 Instructions

2.1 Provided materials

For this lab, some skeleton code has been provided to help you create your solutions. You can download this in a folder called `lab2robotcontrol.zip` from the course wattle page.

Apart from that, to help you with line following, we also provide a short tutorial of how to seek a dot. There is an example script that can give you some ideas of how to follow the line. You can download this in a folder called `Lab Tutorial-Seek a Dot` from Wattle.

2.1.1 Recommended Directory Layout

You should have a directory `ENGNX627_code` from the previous lab. The provided code is designed to work when put in the following directory structure.

```
ENGNX627_code
├── simulator
├── lab1kinematics
│   └── ...
└── lab2robotcontrol
    ├── calibrate_parameters.m
    └── follow_line.m
```

The remaining lab folders will be added in a similar fashion as they are made available in the semester.

2.2 Parameter Calibration

As discussed in lab 1, the robot operates using wheel velocity commands of the form (w_L, w_R), where both w_L and w_R are integers in the range -100 to 100.

In lab 1, you had to solve for the necessary wheel commands given linear and angular speed commands of the form (u, q). This solution depended on two parameters: the wheel track T and the scale parameter γ, which were provided.

In this lab, you need to come up with a procedure to calibrate the parameters T and γ by driving your robot and measuring its position and angle at the start and end.

Task Open the provided skeleton script `calibrate_parameters.m` and implement a procedure to estimate the robot's parameters T and γ using linear regression. Linear regression is revised in week 4, but you can get started with the code now and add the regression code next week.

Hint Begin by focusing on the scale parameter γ. Recall from lab 1 that the linear velocity of the robot is given as $u = \frac{\gamma}{2}(w_L + w_R)$. Now, suppose you set $w_L = w_R = 50$ and drive the robot forward for 5s. If you measure the distance the robot has driven, you can calculate u. From here it is possible to solve for γ using the equation for u in terms of w_L and w_R.

Question 1.a Describe the experiment process you undertook to calibrate the robot.

Question 1.b Present your calibration results.

Question 1.c Explain how you applied linear regression to get your results, using diagrams where appropriate.

Question 1.d Compare the quality of your integrated kinematics from before and after the calibration procedure. How does an incorrect scale parameter affect your estimate of the robot motion? What about an incorrect wheel track?

2.3 Line Following

The goal of this part of the lab is to make your robot follow a line on the ground, and stop at specially marked points along the line.

2.3.1 Warm up

We provide some supporting materials to help you get started. In the lab tutorial, you will learn how to seek a dot and drive your robot towards the dot. There's a script `seek_dot.m` which shows you how to do this. Line following shares a similar algorithm design.

2.3.2 Line identification

Before you can make your robot follow a line, you will need to find the line in the image. One way is to use image thresholding. The line should always be roughly in front of the robot, so you can choose to use only the bottom part of the image. For example,

```
img = img(end-30:end, :);
```

From here, you can use the matlab command `im2bw` to identify only the line in the image. If you now compute the average x coordinate of the pixels of the line, this gives you the x coordinate of the line. Now you just need to normalise this coordinate to use it in line following (make sure the coordinate is always between -1 and 1).

Task In the script `follow_line.m`, program a method to determine appropriate velocity commands based on an image received from the robot. You may use the method suggested above, or develop your own. You may find that you are able to improve the performance by using some clever image processing!

2.3.3 Line Following

Your final task for this lab is to make your robot follow a line. Your solution needs to be written in the script called `follow_line.m`. In this script, you will need to place your robot at the start of the line, and then make it follow a line. The steps in following the line are:

1. Obtain an image from the robot
2. Identify the line
3. Determine if the end of the line has been reached. If so, break the loop.
4. Send an appropriate velocity command to make the robot continue driving on the line
5. Repeat

Task Complete the script `follow_line.m` as described above.

Question 2.a Describe your line following algorithm, using equations, code snippets and diagrams where appropriate.

Question 2.b Present the results that demonstrate the line following performance.

Question 2.c While following a line segment, integrate your robot's kinematics. Plot the trajectory of your robot based on these kinematics and compare it with the true shape of the line you followed. (Integrated kinematics is not the true trajectory that the robot moves on.)

Question 2.d How do different shapes of line affect the error in the robot's trajectory?

2.3.4 Extension Work: Error Ellipse

From the linear regression, it is possible not only to determine your robot's calibration parameters, but also to approximate the standard deviation error of those parameters. Using this measurement of the error, you can estimate over time how confident you are in your robot's position. The extension task is to

find an estimate of the error, and to draw error ellipses on the plot showing the trajectory of your robot (and provide a generative noise model - that is a formula) for the odometry model for your robot for straight line motion and for curved motion. You can check online resources on how to plot error ellipses from covariance matrices.

This part doesn't count in the page limit, so feel free to add figures and equations. We don't give mark to the extension work, but if you provide answers in your report, we can give you feedback on this.

3 What to hand in

For this lab, you only need to submit a lab report addressing the questions in this lab sheet. The report must titled `u1234567_engnX627_Lab2report.pdf`, and must be in the `pdf` format. The report structure should be

```
u1234567_engnX627_Lab2report.pdf
├── Title
├── Q1
│   ├── Q1.a
│   └── ...
├── Q2
│   └── ...
└── Appendix
```

The answers to these questions in the report must not exceed 4 pages in length. The page limit doesn't include the Extension work and Appendix, so you can add more figures and code snippets into your appendix. If you are not familiar with writing an engineering report, refer to the video `How to write a good report` on Wattle.