EE346 - Mobile Robot Navigation and Control
(DRAFT)

Winter 2022
Laboratory #2 (5%)
Date: TBA
TurtleBot 3 and Line Following

Objectives

- Become familiar with using the Gazebo simulator to program a robot
- Study how launch file works.
- Study the basics of image processing and computer vision by implementing a simple line following algorithm

Procedure

In this lab, you will begin to use a simulated TurtleBot3 and build a non-trivial ROS application for TurtleBot3. You use the ROS simulator of Gazebo to simulate the operation of TurtleBot3. The application in this case will be for TurtleBot 3 to follow a line on the floor with help of a camera on the robot. Three different line-followers will be implemented, two from revised version of open-source code and the third being an improved version of one of the first two.

Part I: Line Following in Gazebo in C++

In this part of the lab, you will download and reproduce the line following application available at https://github.com/sudrag/line_follower_turtlebot. In this application, the simulated robot follows a yellow line painted on the floor. Study the C++ implementation of the application first. To install the package and run the application, please follow the instructions in the document "Lab 2 Instructions.pdf" that is available on course Blackboard and posted in the course QQ group. The installment process on your computer is slightly different from the online description because the original code was developed for Ubuntu 16.04 whereas we use Ubuntu 18.04. Demonstrate and explain the operation of your simulated line following robot to the TA while showing the ROS graph you generate with the help of rqt_graph.

In the "To run" section of the github site above, you use a launch. Locate the lf.launch in the directory ~/catkin_ws/src/line_follower_turtlebot/launch, and study its contents with the help of http://wiki.ros.org/roslaunch/Tutorials/Roslaunch%20tips%20for%20larger%20projects. Match the nodes that are "launched" by lf.launch, with those you see the ROS graph when you run rqt_graph.

Once you get the C++ line-follower to work, download the new Gazebo world model with a loop in white available on the course Blackboard, in a file called Lab2WhiteLine.zip. First save a copy of the current folder "Maps", and then replace the files in Maps/ with files in Lab2WhiteLine.zip. There is no need to change the file names. Note that although the new files define a white line,

the name of the folder "yellow line" is not changed in order for the launch file to work. Revise the original line follower node, `linedetect.cpp` specifically, so that your simulated TurtleBot 3 is able to follow the white line. (Hint: you only need to modify the two 3-vectors that define the ranges of HSV values that represent white.

Part II: Yellow Line Follower with Python

In this part of the lab, you will experiment with accomplishing the same task as Part I but with a node written in Python. Please refer to the page http://edu.gaitech.hk/turtlebot/line-follower.html. Study the program, line_follow.py, written in Python. Follow the instructions in "Lab2 Instructions.pdf", beginning at "With Python Script", so that your simulated TurtleBot3 can follow the yellow line as in Part I (upon restoring the original files in Maps). Compare the two yellow-line followers in C++ and Python, learn how to set up ROS nodes to perform this task, and determine the algorithmic differences between the two line-followers. In particular, pay attention to how the yellow color is defined and how the error is corrected in the two cases. Demonstrate and explain the operation of your simulated line following robot to the TA while showing the ROS graph you generate with the help of rqt_graph.

Part III: Optimization of Line Following

In this part of the lab, you will construct your own line follower by modifying either the C++ or the Python program in the previous two parts in order to improve the line following performance in terms of running time. You can consider changing how the line is detected and how the error is corrected, and there is no standard solution. We will measure the performance of your solution in terms of the percentage by which the running time has been improved with respect to the lower of the two running times in Part I and II, in order for the performance improvement to be independent of the computer that you use to complete this assignment. The performance among all the students will be ranked according to the percentage of improvement, and the mark students receive in Part III will depend on their ranks.

Marking Guide

Due date is to be determined (TBD).

Part I: 30%
Part II: 30%
Part III: 40%