CS/ME/ECE/AE/BME 7785

Lab 2

Due: September 21, 2018 at 3 pm

Overview

The objective of this lab is to get you familiar with writing ROS code for the Turtlebot 3. In this lab you will create your own ROS package containing two communicating nodes that enable the robot to turn in place to follow a ball. You will also create one debugging node that allows your computer to display the processed image from the Turtlebot. You can use your image processing code from Lab 1 that enables the robot to identify the location of a ball in the frame.

For this lab, there can be severe issues with lag when transmitting the images over Wi-Fi to each person's individual computer. For this reason, you can debug by transmitting images from the Turtlebot to your personal computer but once your parameters are set, you should run the roscore and all files on-board the robot. You *should not* have images passed from the Turtlebot to your laptop for processing during the demo. To move folders from your computer to the robot, use the scp (secure copy) command. For example,

scp -r <Directory on your cpu to copy> burger@<ip-of-burger>:<Directory to copy to on robot>

Since the code will be run on the robot to avoid network lag issues, please name your node `LastName_ball_follower` to avoid overwriting others' code. When the lab is finished, or you are finished testing, please remove your code from the robot.

We strongly encourage you to use all available resources to complete this assignment. This includes looking at the sample code provided with the robot, borrowing pieces of code from online tutorials, and talking to classmates. You may discuss solutions and problem solve with others in the class, but this remains an individual assignment and each person must submit their own solution. Multiple people should not jointly write the same program and each submit a copy, this will not be considered a valid submission.

Lab

For this lab create a new package called **LastName_ball_follower** (refer back to the ROS tutorials from Lab 1 if needed). Some potentially useful dependencies include `rospy`, `roscpp`,

`sensor_msgs`, `std_msgs`, `geometry_msgs`. Add two nodes to the package:

find_ball: This node should subscribe to receive images from the Raspberry Pi Camera on the topic `/raspicam_node/image/compressed`. Example code in Python:

```
img_sub = rospy.Subscriber("/raspicam_node/image/compressed",CompressedImage,callback)
```

From this point you should be able to use the same code from Lab 1 to take this image and identify the location of your ball in the frame. Once you've located the ball in an image, this node should publish the pixel coordinate of the ball. To do this, you can create your own message type (refer to ROS tutorials) or use an existing message type such as the `geometry_msgs/Point` message. To help debug this, it may be useful for this node to also publish the processed frame for a node run on your computer to subscibe to and display.

drive_wheels: This node should subscribe to the ball coordinate messages from your **find_ball** node and publish velocity commands to the robot:

```
pub = rospy.Publisher('/cmd_vel', Twist, queue_size=5)
```

Configure your code to have the robot turn to follow the ball in the image (turn right when the ball is on the right of the image, turn left when it is on the left, and stand still if the ball is directly in front of the robot). The robot should not drive forward or back, only turn.

We will provide balls of several sizes and colors for your use in the lab. Feel free to use any of them, or a different ball/circular object of your choosing. For this lab, the ball can be at any distance from the webcam that you choose (not taking up the entire frame).

Grading Rubric

Successfully run all code^o on the robot	25%
Robot rotates in the direction of the ball consistantly*	50%
Communicate information between your two nodes	25%
Code left on the robot	-10%

^{o} All code does not include any debugging code you write for your personal computer.
* Consistantly means it does not rotate the wrong direction due to noise/image delay often, the robot will get false and noisy readings occasionally but we will not deal with these problems in this lab.

Please do not leave your code on the robot after demoing or testing as it may interfere with others' tests. Note the penalty in the grading rubric.

Submission

Lab submission consists of two parts:

1. Your ROS package and any supplimentary files, in a single zip file called YourLastName_YourFirstName.zip uploaded on Canvas under Assignments–Lab 2.

2. A live demonstration of your code to one of the course staff by the date listed at the top of this page. We will set aside class time and office hours on the due date for these demos, but if your code is ready ahead of time we encourage you to demo earlier (you can then skip class on the due date).