CS/ECE/ME/BME/AE 7785

Lab 2

Due: Monday September 11th, 2017 3pm

Overview: The objective of this lab is to get you familiar with the robot and to get started writing code in ROS. You will accomplish two things: 1) run provided Turtlebot3 code to get familiar with running the robot, 2) create your own package containing two communicating nodes that enable the robot to turn in place to follow a ball. Note that the first part will take far less time than the second, so budget your time wisely.

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 Instructions:

First, make sure your ROS setup is working and you are able to run code on the robot. All instructions for the Turtlebot3 are available online here. To test your setup:

a. Run the Bringup instructions (listed as "8.Bringup" on the left side of the above website) Note that you may have to refer to parts of "5. PC Software Setup" the first time you do this. On the robot itself, edit the `.bashrc` file such that the IP address listed under ROS_MASTER_URI is the IP of the laptop where you are running roscore. Don't forget to `source ~/.bashrc` whenever you edit that file to make the changes take effect.

b. Once you run Bringup, run the "9. Teleoperation" demo using your keyboard to control the robot

c. (Optional) If you want to test out mapping and autonomous path planning, run "10. SLAM" and "11. Navigation".

d. Run the TurtleBot Follower Demo under "15. Applications"

For the second half of the lab, create a new package called `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)
```

Upon receiving an image, your code should convert it to OpenCV format:

```
cv_image = bridge.compressed_imgmsg_to_cv2(data, "mono8")
```

And then process the image to look for circles. An easy place to start is to use the HoughCircles() functionality within OpenCV:

```
circles = cv2.HoughCircles(cv_image, cv2.HOUGH_GRADIENT, 1, 90, param1=70, param2=60,
                           minRadius=50, maxRadius=0)
```

As described in class, this alone will likely not be enough. Experiment with ranking the returned circles based on color or other properties. Normalizing and slightly blurring the image can also improve the performance of HoughCircles().

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.

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 of your choosing. For this lab, the ball can be at any distance from the robot that you choose.

Grading Rubric:

Successfully run code on the robot	15%
Find ball in image >50% of the time*	35%
Communicate information between your two nodes	15%
Drive the robot	35%

*We're setting a pretty low bar for ball detection here because there's already a lot going on in this lab and because we'll talk about ways to improve detection later in the semester.

Submission: Lab submission consists of two parts:

1. Your ROS package, 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 11th 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 11th).