CS/ECE/ME/BME/AE 7785
Lab 3

Due: October 5^{th}, 2018. 3pm

1 Overview

The objective of this lab is to explore PID control and combining multiple sensor inputs. The goal is to have the robot chase a desired object observed in its local coordinate frame. Specifically, this means you must make the robot always face the object and maintain a desired distance from the object. You can track any object you would like as long as it can be moved by one of the instructors. If you want, you can use the ball tracking script you developed in previous labs. We have seen in class that it is not possible to determine an objects distance from a robot at all times from camera data alone so you will have to use the LIDAR as well.

As you did with lab 2, you should run the `roscore` and all files on-board the robot. 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_chase_object` to avoid overwriting others' code. When the lab is finished, 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.

2 Lab Instructions

Create a package called `LastName_chase_object` (refer back to the ROS tutorials from Lab 1 if needed). Similar to the last lab, useful dependencies include `rospy`, `roscpp`, `sensor_msgs`, `std_msgs`, and `geometry_msgs`. You can add as many nodes as you like. An example structure would be:

detectObject: This node should be similar to the `find_ball` you created in last lab. It should subscribe to the raspberry pi camera node and publish the location of the center of the object you're tracking.

> Note: unlike previous labs, you will now be coordinating your camera with your LIDAR so they will have to agree on reference frames and units. This means you should convert your pixel values to degrees or radians and know the relation between the reference frame of your camera and your LIDAR.

getObjectRange: This node should subscribe to receive the location of the object from your `detectObject` and publish the angular position and distance of the object using some combination of the LIDAR data and camera data. To get the LIDAR data, this node should also subscribe to the `\scan` topic.

LIDARS produce a lot of data and are one of the more complicated sensors to use. For more information on the LIDAR used on the turtlebot3 please look here (http://emanual.robotis.com/docs/en/platform/turtlebot3/appendix_lds_01/). You can also echo the publisher to view the data that is being produced.

chaseObject: This node should subscribe to the data being published by `getObjectRange`. Based on this data you should create two PID (or some variant) controllers to follow your tracked object at a desired distance (meaning the robot drives forward and backwards if needed). One controller should act on the angular error to make the robot face the object while the other acts on the linear error to make the robot maintain a distance from the object. This node should publish the velocity commands for the robot to follow in the same manner you did in lab 2.

> Note there are PID controllers already acting on the motors of the turtlebot3. If this was a custom robot, or you did not like the motor controllers of the robot, you would have to design a PID controller for the motors of the robot as well that would have a settling time much faster than your high level controller!

3 Questions

These questions must be answered in a writeup submitted with your code. This does not have to be a formal lab report, just answer the questions directly. Include supplemental material where you think it would help. These questions should also be considered while developing your code. You may be asked to answer these questions in person during the demo.

1. What is the sampling time of your system? How/Why did you choose this (how fast can you get sensor data/how fast is your reference signal changing)? Why do you need a constant sampling time?

2. What variant of PID control did you use? Why? What is your controller susceptible to?

3. If you use an integral term, how do you deal with windup? If you use a derivative term how do you deal with noise/fast changes in the object's location? If you just used purely proportional control, how do you deal with disturbances?

4. What happens if you make your gains too high? Why does this happen?

5. What does it mean for this system to be unstable? A helicopter/plane will fall out of the sky if it uses an unstable controller, what does your robot do when your controller is unstable?

6. Describe your algorithm to determine where the object is relative to the robot. Include mathematical expressions used.

4 Grading

Consistently determine the location of your object (distance and angle relative to the robot)	25%
Spin the robot to face the object	25%
Drive the robot to maintain a distance from the object	25%
Question responses (in person and write up)	25%
Code left on the robot	-10%

5 Submission

You have two required submissions for this lab.

1 Your ROS package and write-up, in a single zip file called `LastName_FirstName.zip` uploaded on Canvas under Assignments–Lab3.

2 A live demonstration of your code to one of the instructors. This can be done anytime before the due date at the top of this lab. Class will meet in the lab room on the due date to allow everyone to demo their controllers. If you demo before the due date you do not need to come attend class that day.