CS/ME/ECE/AE/BME 7785

Lab 3

Planned ROS Computation Diagram Drawing Due: February 14, 2025

Lab Demo Due: February 21, 2025 at 4 pm

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 object 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.

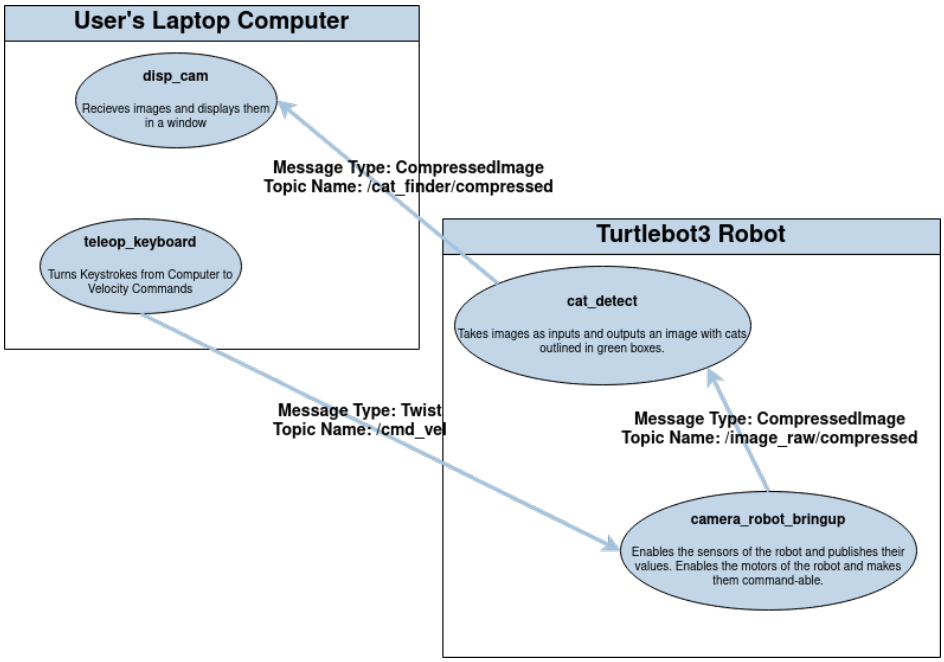
Unlike in lab 2, you are no longer required to run all files on-board the robot. If you wish to run some computational ROS nodes on your computer, you can choose to. Be aware that WiFi communication can be unreliable. If you choose to constantly communicate information between the robot and your computer you are accepting the risk that your demo may fail due to WiFi communication.

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 a team assignment and each team must submit their own solution. Multiple teams should not jointly write the same program and each submit a copy, this will not be considered a valid submission.

Planned ROS Computation Diagram Drawing

Before you start coding, it is often a good idea to sketch out how and where you will do computation and send data to solve the problem.

The first step of this lab is to draw a ROS computation graph of how you will solve the problem. This should look similar to an rqt-graph but with a few more details. You should include the nodes and what they are computing, the topics and messages they subscribe and publish to, and where the nodes are physically going to run (i.e. the Turtlebot or your computer). For example, if I were to make ROS computation graph for tele-operating the robot while receiving a camera feed that is finding cats, it may look like the following:



You can draw this diagram on the whiteboard or in your notebook and take a picture for submission or use an online tool like draw.io. The purpose of this step is to have you plan your software solution before coding furiously!

2 Lab Instructions

Create a package called `TeamName_chase_object` (refer back to the ROS tutorials from Lab 1 if needed). For your `Team Name` you can make one up or use the `Team Number` you were assigned with your lab teams. Note, package names must begin with a letter so if your `Team Number` was 1 you could name your package `team1_object_follower` or `one_object_follower`. Similar to the last lab, useful dependencies include `sensor_msgs`, `std_msgs`, and `geometry_msgs`. You can add as many nodes as you like. An example structure would be:

detect_object: This node should be similar to the `find_object` you created in last lab. It should subscribe to the raspberry pi camera node and publish the location of the 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.

get_object_range: This node should subscribe to receive the location of the object published from your `find_object` node 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. You can echo the topic to view the data that is being produced. For more information on the LIDAR used on the Turtlebot3 please look at:

http://emanual.robotis.com/docs/en/platform/turtlebot3/appendix_lds_02/

For more information about the LaserScan message type please look at:

http://docs.ros.org/en/noetic/api/sensor_msgs/html/msg/LaserScan.html

chase_object: This node should subscribe to the data being published by `get_object_range`. 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.

1. What is the sampling time of your system(hint: how fast can you get sensor data)? Why does sampling time matter, even with a subscriber based controller?
2. What variant of PID control did you use? Why?
3. If you use a proportional controller (P) why does the robot achieve its steady state (distance from the object) when disturbances like friction from the floor, wheel shaft, or even Sean's finger on the tire are present?

   If you use an integral term in your controller (I, PI, PID), what is integral windup and how did you reject it in your controller's realization?

   If you use a derivative term how does your controller implementation deal with noise/fast changes in the object's location that would cause a large derivative term?
4. What does it mean for a system to be unstable? What behaviors will your robot display if it uses an unstable controller?
5. Describe your algorithm to determine where the object is relative to the robot. Specifically, how do you use the camera and LIDAR data to produce a desired velocity vector? Include mathematical expressions used and supportive figures where appropriate.

4 Grading

The lab is graded out of 100 points, with the following point distribution:

Planned ROS2 Computational Diagram	20%
Rotate the robot to face the object	20%
Drive the robot to maintain a distance from the object	20%
Robot stops at desired distance and heading of the stationary object within 5 seconds	10%
Question responses	30%

5 Submission

1. Perform a live demonstration of you running the robot to one of the course staff by the listed deadline.
2. Put the names of both lab partners into the header of the python script. Put your python script, your writeup, and any supplimentary files, in a single zip file called Lab3_LastName1_LastName2.zip and upload on Canvas under Assignments–Lab 3.
3. Only one of the partners needs to upload a submission.

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 in any of the office hours (you can then skip class on the due date). Office hour times are listed on the Canvas homepage.