Warmup Project

Objectives

Modified from CMSC 20600

Your goal in this project is to get comfortable programming the Turtlebot3 robot in the Gazebo simulator. You'll be asked to program the robot to perform one or more behaviors. Additionally, in the process, you'll learn about tools and strategies for debugging robot programs. If you find yourself stuck on something, email jason27146913@gmail.com or join the TA office hour.

Learning Goals

- Get comfortable with ROS
- Brush up on your Python programming
- Learn about processing sensor data from the robot
- Learn about programming robot behaviors
- Learn ways to debug robot programs
- Learn about reactive robot control

Logistics

This assignment is to be completed individually. You can get help from your fellow classmates setting up your programming environment and completing the ROS beginner tutorials. For the meat of this problem set (programming Turtlebot3 behaviors), you can ask your fellow classmates for general advice or tips on ROS/robot programming. However, the work of programming the robot's behavior must be done by you (no sharing code, no copying code, no looking at other people's code).

Getting Started

Before you're ready to program your robot, you'll need to take the following steps:

- Follow the instructions on the VM import tutorial to set up your programming environment for programming the Turtlebot3 robot.
- Check out our guide for using the Gazebo simulator on the Gazebo Simulator resource page on our course website for tips on how to use the simulator.
- Read through the ROS Resource page for a quick reference to useful ROS commands, debugging using RViz, and instructions on how to record and use rosbags.
- Put the `lab0_warmup` folder within your `~/catkin_ws/src/` directory (where ROS packages should be located).

Running Your Code

In order to run your code for this project, you'll likely have at least 3 terminals running: one running `roscore`, one used to launch and run the Gazebo simulator, and one running the program you've written to control the robot. Here's an example of what the commands used to run these three tasks may look like.

First terminal: run `roscore`.

```
$ roscore
```

Second terminal: launch and run the Gazebo simulator with a specific world file. For the wall follower, you'll use the `turtlebot3_in_room.launch` launch file I've provided in the `lab0_warmup` ROS package. For the other robot behaviors, you

can use the `turtlebot3_empty_world.launch` launch file in the `turtlebot3_gazebo` ROS package.

```
$ roslaunch [package-name] [launch-file-name]
```

Third terminal: run your program to control the robot. For example, if my robot wall following code is in the `scripts/wall_follower.py` file, my package name is `lab0_warmup` and my script name is `wall_follower.py`.

```
$ rosrun [package-name] [name-of-your-script]
```

Note: You may have to run `chmod u+x wall_follower.py` (or substitute `wall_follower.py` for the name of your python script) in order to make your python script executable.

Turtlebot3 Behavior Programming

Driving in a Square

Your goal is to write a ROS node that commands the Turtlebot3 robot to drive a square path. You can accomplish this by using either timing (e.g., move forward for a fixed amount of time then make a 90 degree turn four times) or using Turtlebot3's odometry (check out the `/odom` topic). You can choose which approach to take, however, we will point out that using timing is significantly easier. The following are good examples of what you can expect your drive in a square behavior to look like:

Demo1
Demo2
Demo3

Tips:

- One thing you should notice right away is that there is **noise** in the environment and in the mapping of your commands to the robot and how it executes them. Your robot will not perfectly execute your commands due to friction, imprecise elements of controlling physical motors, etc.
- Your driving in a square does not have to be "perfect". It should *roughly* drive in a square. If it ends up a few inches away from where it started, that's more than fine.
- If you are spending hours on this, hitting mental walls, or feeling frustrated, **reach out to TA for help!** This is your first time programming a robot, give yourself some slack, and reach out for help. That's what we're here for!

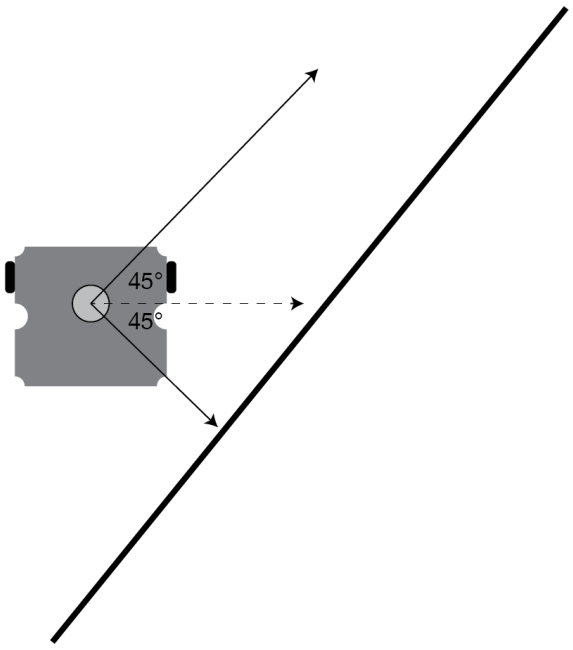
Wall Follower

Your goal with this robot behavior is to 1) have your robot navigate close to a wall, 2) drive alongside the wall at an approximately fixed distance, and 3) handle corners. You'll use the `turtlebot3_in_room.launch` launch file, where the Turtlebot3 is situated in a square room. When working properly, your robot should run around this square room (either clockwise or counterclockwise) indefinitely. Here are some examples of what your wall follower implementation may look like:

Demo1
Demo2
Demo3

Tips:

- Your robot should be able to follow walls and round corners in any environment with a wall and shouldn't be specifically programmed to follow walls in the square room provided (if we drop your robot anywhere in this square room, it should be able to find a wall and start following it).
- You may find this diagram helpful when thinking about what the goal should be of your robot controller:



Deliverables

You'll submit this project on CGU Teams (both the code and rosbag files).

Code

Please put all of your code in a `scripts` directory within `lab0_warmup`.

Rosbags

For each of the behaviors you program for your robot, we ask that you record your robot performing that behavior in a rosbag. **Please record only the `/scan` and `/cmd_vel` ROS topics so that your bagfiles aren't enormous.** Please put all of your rosbags in a `bags` directory within `lab0_warmup` with a name that will help us easily identify which behavior you've recorded. For ease of use, here's how to record a rosbag:

```
$ rosbag record -O [filename.bag] [topic-names]

example: rosbag record -O wall_follower.bag /cmd_vel /scan
```

For more information on rosbags, please refer to the ROS Resources page.

Here is a list of the files that you are likely to make/edit during this project:

```
lab0_warmup/bags/wall_follower.bag

lab0_warmup/bags/drive_square.bag

lab0_warmup/scripts/drive_square.py

lab0_warmup/scripts/wall_follower.py
```

Grading

The warmup project will be graded as follows:

- 40% ROSbag Recordings
  - 20% Driving in a Square Rosbag Record
  - 20% Wall Follower Code Rosbag Record
- 60% Code
  - 30% Driving in a Square Code
  - 30% Wall Follower Code

Submission Deadline and Method

Language : python3

Deadline : Thu 02 Mar 2023 13:00 (UTC+8)

Delay policy : One Day -> points * 0.7
Two Days -> points * 0.5
After Two Days -> 0 point

Everyone has a chance for a one-day late submission this semester!!!

Submission Method:
Upload a zip file to CGU Teams with the format :

```
m1234567_lab0.zip
  + m1234567_lab0
      - scripts
          + wall_follower.py
          + drive_square.py
      - bags
          + wall_follower.bag
          + drive_square bag
```

**Do not include other files other than those mentioned above*

All file names should be in **lowercase and **only zip files** (No rar and 7zip)*

****Incompatible format will not be graded.***

Notice : Zero points for plagiarism!!!

(either from the internet or copy from classmate)