PS1: Open Loop Control of Turtlebot

Sasha Belotserkovskaya, avb27@case.edu; Kristina Collins, kvc2@case.edu; Shawn Qian, xxq82@case.edu; Connor Wolfe, cbw36@case.edu

I. Introduction

This assignment presents an example of open-loop control in the ROS STDR (Simple Two-Dimensional Robot) Simulator. It serves to demonstrate creating and editing a ROS package, running STDR, taking a screencast and submitting it, and exploring the limits of open-loop control.

II. Testing

A. Demonstration Steps

First, connect to Turtlebot using the instructions in the lab assignment. The Turtlebot acts as the ROS master. Recommended to use Google's DNS for stability. Also, note that connecting to the Turtlebot in one terminal does NOT connect you to it in other terminals. To wit, for deeplearning02:

- **host deeplearning02w 8.8.8.8** Take note of the robot's IP, using Google's DNS.
- **ifconfig** Take note of the computer's IP.
- **ping deeplearning02w 8.8.8.8** Check to ensure that data can be exchanged.
- **ROS_MASTER_URI = http://deeplearning02w.eecs.cwru.edu:11311** Set the Turtlebot as ROS master.
- **export ROS_IP=XXX.XX.XXX.XX** Use the computer's IP here.

It's wise to verify connection by checking rostopics with the command **rostopic list**. Next, run the program:

- **roscd**
- **rosrun bacon1 bacon_commander**

The Turtlebot should start to move in a square path.

B. Tips and Notes

It's wise to keep an eye on the Turtlebot and make sure it doesn't bump into anything. The Turtlebot deeplearning03 connected at first, but tended to disconnect intermittently; we believe this may be the result of the Case DNS having trouble. For that reason, the instructions above and in the README recommend using the Google DNS and addressing the turtlebots by IP rather than hostname. We demonstrated our code on deeplearning02. Finally, note that connecting to the Turtlebot in one terminal does not create a connection in other terminal windows.

III. Notes and Evaluation

As shown in our video, we ran a qualitative test of the code, using the floor tiles in Glennan 210 to demarcate distances. Our robot was successful in returning to its starting point, although it did so with a slight shift in position and angle. This error could, perhaps, have been reduced through extensive manual tuning of the timing in the code, but this method would have been inferior to a closed-loop solution.

A. Quantitative Methodology Results

To quantify the repeatability of the open-loop code, a series of tests were run in which the robot was centered on a marked block. The code was then run, the error quantified, and the robot replaced in its original position for the next test. The results of the testing are listed in the table below.

The shifts in X and Y are in units of floor tiles; θ is in units of degrees, with clockwise defined as positive.

Test	X Shift	Y Shift	θ Shift
1	.25	-.25	+45 deg
2	.25	-.3	+50 deg
3	0	-.5	+45 deg
4	.5	-.5	+45 deg
5	.4	-.5	+45 deg

IV. Conclusion

On average, the position is reliably accurate to within a floor tile, which is pretty good for our purposes. The most concerning error is the shift in θ, which would cause the robot's path to vary widely if it were commanded to move in a loop without correction. This error and the position errors are cumulative, so putting an open loop controller like this in a while loop is a generally bad idea.