Institute for Aerospace Studies
UNIVERSITY OF TORONTO

LABORATORY II

Pose-to-Pose Control of a Robot

ROB301 Introduction to Robotics
Fall 2020

1 Introduction

For a robot to interact with its surroundings, it must be able to move within them. The focus for this, the second laboratory exercise in the course, is pose-to-pose control for the locomotion of a robot. It is the most basic form of robotic control. In this lab, we will only be looking at feedforward or open-loop control. Naturally, we'd expect to find some shortcomings in this approach and it will be your task to investigate them. *You will be required to submit your results to the TAs.*

2 Objective

The objective of this laboratory exercise is to study feedforward or open-loop pose-to-pose motion control of the Turtlebot 3 Waffle. In particular, you will be required

- ▷ *To study pose-to-pose control using a straight-line path*
- ▷ *To study pose-to-pose control using waypoints with straight-line segments*
- ▷ *To study pose-to-pose control using curved paths*

One of the advantages of ROS is that it abstracts away the direct interfacing with the motors. Thus as studied in class you can use speed v and angular rate ω to command the robot.

You will explore different means of achieving pose-to-pose locomotion, including on-the-spot rotation, motion using waypoints, straight-line motion, and motion along curved paths. By the end of the lab, you should be comfortable with coding the Waffle to move to an arbitrary location and appreciate the drawbacks of motion control without feedback.

3 Lab Deliverables

To demonstrate functionality, each team is required to submit a one-page report. In the report, you need to describe how your robot completed all the tasks, and what was your approach. If you were unable to complete the task, explain why that was the case.

Each team is required to submit 3 rosbag files along with the report, where each file records a successful run of each task. You can also submit one additional rosbag file for the bonus challenge. Please refer to the **bold text** in "Lab Deliverables" for instructions on how to start rosbag recordings.

4 Equipment & Software

As introduced in Lab I, the hardware platform used for this lab is the TurtleBot 3 Waffle Pi, and Lab II tasks are very similar to those in Lab I. You will find the starter code in

`catkin_ws/src/rob301_simulation/nodes/lab02.py`

As a quick reminder from Lab I, the following commands should be run in separate terminal windows sequentially: (1) run the ROS launch file for lab02, (2) start Gzweb and visualize the graphical interface on your local browser, and (3) run your custom `lab02.py` node (you should keep these terminal windows open as they are continuously running the program):

`$ roslaunch rob301_simulation lab_02.launch` [Remote PC]

`$ cd ~ /gzweb && npm start -p {port number}` [Remote PC]

`http://100.92.1.X:808X` [Local PC browser]

`$ rosrun rob301_simulation lab02.py` [Remote PC]

5 Assignment

For the purposes of this lab, the robot can be modeled as a "unicycle," whose pose is given by three degrees of freedom: x, y and θ, as measured in a global reference frame. In short,

$$\mathbf{x} = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}$$

Recall that the model is given by

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}$$

The column **x** will be henceforth be formally known as the robot's pose (position + orientation). The goal of this lab will be to command the robot speed and angular rate in order to move from one pose to another.

5.1 Pose-to-Pose Locomotion Using a Straight-Line Path

To demonstrate the basics of locomotion, you are required to command the robot to move from a start pose $\mathbf{x}_{\text{Start}}$ to a goal pose $\mathbf{x}_{\text{Goal}}$. The goal pose will be marked by a yellow tape on the ground plane in the Gazebo simulation (Figure 1). The most direct way is to plan a straight-line path from the start to the goal and have the robot follow it; you may perform on-the-spot rotations as deemed necessary. Take

$$\mathbf{x}_{\text{Start}} = \begin{bmatrix} 0 \\ 0 \\ 0^\circ \end{bmatrix}, \qquad \mathbf{x}_{\text{Goal}} = \begin{bmatrix} 200\text{ cm} \\ 50\text{ cm} \\ 135^\circ \end{bmatrix}$$

Keep in mind that the Turtlebot's linear/angular velocities are limited: Sending command velocities greater than 0.26 $\text{m}\,\text{s}^{-1}$ or 1.82 s^{-1} will cause issues with your odometry estimate since the motor speeds are limited to these values.

Lab Deliverable 1: Before you move onto the next task, you are required to demonstrate functionality by recording the successful run using ROS bag (i.e., robot moves to the goal pose in straight-line path). You can record by running

```
$ rosbag record /cmd_vel /odom
```

prior to running `lab02.py`.

5.2 Pose-to-Pose Locomotion Via Waypoints

Consider now a path based on a series of waypoints. In this demonstration, you are asked to move the robot through a square path, defined by waypoints A, B, C, back to its original pose. That is, let

$$\mathbf{x}_{\text{Start}} = \begin{bmatrix} 0 \\ 0 \\ 0^\circ \end{bmatrix}, \quad \mathbf{x}_A = \begin{bmatrix} 100\text{ cm} \\ 0 \\ 90^\circ \end{bmatrix}, \quad \mathbf{x}_B = \begin{bmatrix} 100\text{ cm} \\ 100\text{ cm} \\ 180^\circ \end{bmatrix}, \quad \mathbf{x}_C = \begin{bmatrix} 0 \\ 100\text{ cm} \\ 270^\circ \end{bmatrix}$$

and $\mathbf{x}_{\text{Goal}} = \mathbf{x}_{\text{Start}}$.

The A, B, C waypoints are marked by blue tape on the simulation ground plane, and the initial pose is marked by a white tape (Figure 1). Note that you will need to relaunch `lab_02.launch` to move the robot back to the original pose. You may need to restart Gzweb and refresh your browser as well.

Lab Deliverable 2: Before you move onto the next task, follow the instructions above to start a rosbag recording of the rostopic `/cmd_vel/odom` **prior to running the custom ROS node.**

5.3 Pose-to-Pose Locomotion Along a Curved Path

Unfortunately, some robots may not be capable of pivoting on the spot. (Consider nonholonomic robots!) In this case, it would be necessary to design a curved path for the robot to follow. Assuming that the minimum turning radius for the robot is 25 cm, design a path that will take it from start to goal as given in Part 5.1.

To accomplish this task, you will need to set the speed and angular rate of the robot to follow the desired path. Once again, estimate the error along the path and at the goal. Try different speeds, and correspondingly different angular rates, profiles and comment on performance.

Lab Deliverable 3: When you have successfully completed this task, record the ROS topic `/cmd_vel/odom` of a successful run using rosbag.

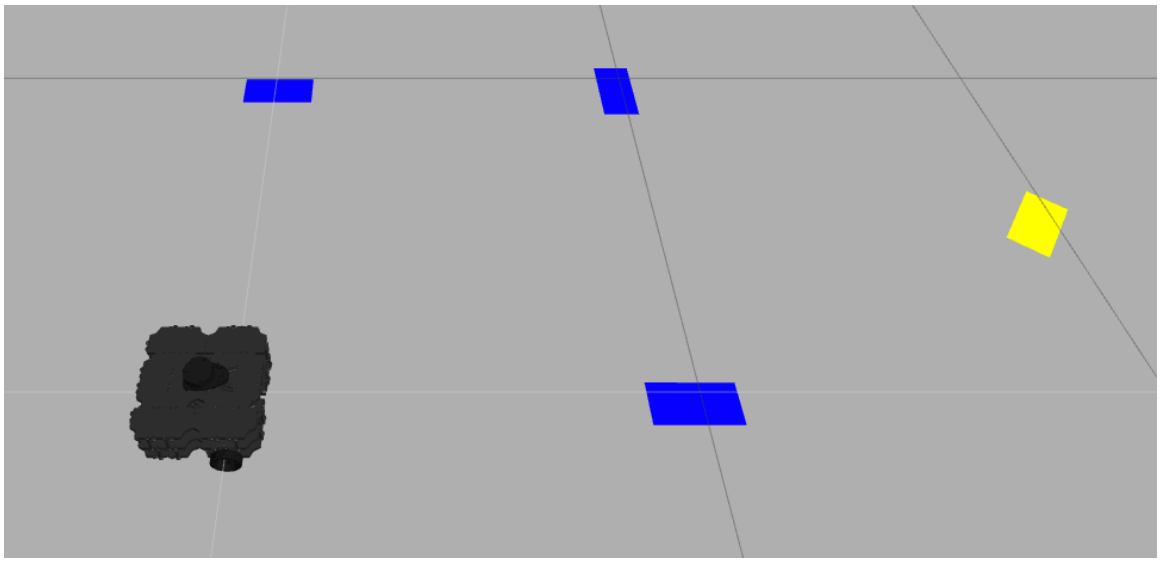

Figure 1: Gazebo Simulation Environment

5.4 Bonus Challenge: A Complex Path

If you've finished the first four assigned tasks, try a more challenging path for the robot. Suppose we'd like the robot to follow a path described by

$$y(x) = 50 \sin\left(\frac{\pi}{25}x\right) \quad \text{[cm]}$$

Try different speeds, and correspondingly different angular rates, profiles here as well and comment on performance.

Lab Deliverable 4 (optional): If you have completed the bonus challenge, you can include a rosbag file for this task in your deliverable.

6 Closing Remarks

In this laboratory exercise, you should have been able to familiarize yourself with feedforward or open-loop control in the context of pose-to-pose control as well as following desired paths. You should also have been able to discern some of the limitations of feedforward control, upon which we will seek to improve by introducing feedback in Lab III.

7 Additional Resources

1. Introduction to ROS, ROB301 Handout, 2020.
2. Robotis e-Manual, `http://emanual.robotis.com/docs/en/platform/turtlebot3/overview/`.
3. "SSH: Remote control your Raspberry Pi," The MagPi Magazine, `https://www.raspberrypi.org/magpi/ssh-remote-control-raspberry-pi/`.
4. Official ROS Website, `https://www.ros.org/`.
5. ROS Wiki, `http://wiki.ros.org/ROS/Introduction`.
6. Useful tutorials to run through from ROS Wiki, `http://wiki.ros.org/ROS/Tutorials`.
7. ROS Robot Programming Textbook, by the TurtleBot3 developers, `http://www.pishrobot.com/wp-content/uploads/2018/02/ROS-robot-programming-book-by-turtlebo3-developers-EN.pdf`.