Foundations of Robotics

Lab 2: Open Loop Control

Overview

In this lab, we are going to move the Turtlebot robot around using open-loop control strategy (i.e. sending commands only; no feedback). The goal is to familiarize yourself with some basic interactions with the robot, and try tuning parameters by hand.

Specifically, the task is to make the robot move in a square shape to visit four waypoints [5, 0], [5, 4], [1, 4] and [1, 0]. Since we have charging dock located at the origin, we will first ask the robot to move forward for 1 meter to avoid this dock. After this, the robot should move forward 4 meters, turn left 90 degrees, move forward again 4 meters, and so on, until going back to [1, 0] point. Note that the robot is supposed to stop at [1, 0] after completing this square movement, and the Python/C++ script should exit gracefully.

Again, to better accommodate beginners, we provide step-by-step instructions and sample code in Python. However, please feel free to use C++ if you are an advanced developer. For those who are not familiar with ROS, please also take this opportunity to catch up.

Submission

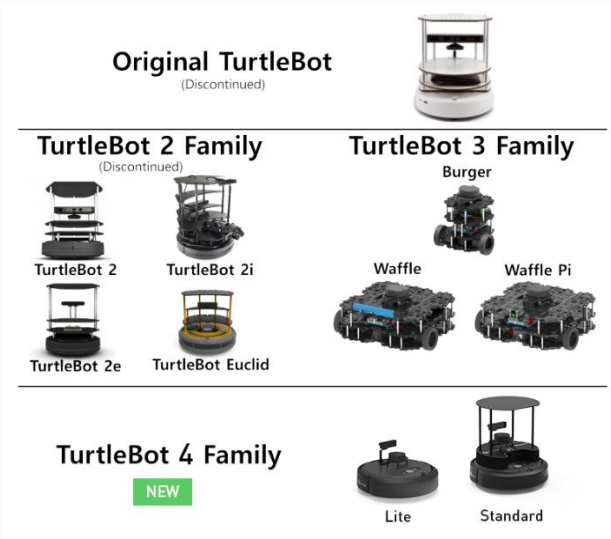
- Files to submit:
  - open_loop.py or open_loop.cpp
  - a few screenshots (with the plot of the square-shape trajectory if possible) to show that you have successfully completed the task
- Grading rubric:
  - The robot can visit all four vertices of the square trajectory (error < 1.0m). Partial credits will be given according to the number of vertices visited.

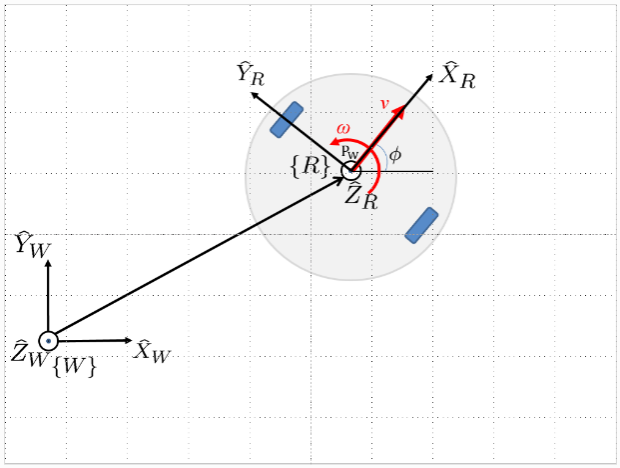
Introduction to Turtlebot

The Turtlebot4 robot is a differential wheeled robot. See the image (left) below for the Turtlebot family.

Though multiple layers of plates/sensors are placed on top of the robot, the kinematics of the robot can be simplified according to the property of its mobile base, which uses

differential drive for locomotion. The differential mobile base has two powered wheels, located symmetrically about its center.





As users, we can send high-level commands (linear velocity `v` and angular velocity `ω`) to the robot. The mobile base will first transform the `v` and `w` commands with respect to the robot center into the desired rotational speed of each wheel, and then control the rotational speed by a feedback control of the current of the motor that drives the wheel.

To describe the position and orientation of the robot, we attach a robot coordinate frame R to it. The origin of this coordinate frame is centered between its powered wheels. The X axis of this frame is pointing forward (along the direction of the linear velocity `v`), the Y axis is pointing to the left, and the Z axis is pointing up.

To track the position and orientation of the robot, we generally define a world reference frame W, in the same plane where the robot moves. With this frame assignment, the robot's position is constrained to the X − Y plane of frame W. Moreover, any rotation between the robot and the world frames can be expressed as a rotation about Z axis. Therefore, the position of the robot with respect to the world reference frame will have the form:

$$P_W = \begin{bmatrix} x \\ y \\ 0 \end{bmatrix}$$

while the rotation matrix expressing the orientation of the robot frame with respect to W will be of the from:

$$R_{WR} = \begin{bmatrix} cos(\phi) & -sin(\phi) & 0 \\ sin(\phi) & cos(\phi) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Programming Tips

1. We follow ROS conventions to use SI units. (i.e. length in meter, time in second, angle in radian). See ROS Wiki article REP 103 Standard Units of Measure and Coordinate Conventions for more information.
2. When a new robot is spawned, the forward heading direction is the positive x axis; the leftward direction is the positive y axis; and by right-hand rule, z axis upward. This is also specified in REP 103.
3. Python is an indent-sensitive programming language, as opposed to C/C++.
   - You can use either `space` or `Tab` for indent, but please do not mix them in one file. Otherwise, you will see syntax errors. (In my opinion, `space` is recommended.)
   - A better way to organize indent is to use an Integrated Development Environment (IDE) for programming, where `Tab` key can be automatically converted into 2 or 4 spaces.
4. The recommended IDE in Linux is VS Code. Just go to the official website, download `.deb` file and install it.
   - After installation, you can go to Extensions and search for ROS. Adding this extension can help you highlight the code and bring more convenience.
5. In Gazebo, you can use `Ctrl + R` to set the robot back to the origin without the need to relaunch.
6. In this lab, you need to finely tune the parameters for open-loop control.
   - Please note that parameters may vary from platform to platform. In other words, the parameters work in your VM may not necessarily work in other machines.
   - In Gazebo, you can take the visualization as feedback (the grid size of the ground is 1 meter) to tune the parameters.

Sample Code

A sample code is provided to make the robot move forward for a certain distance. You need to make changes under `run` function to complete the square trajectory.

- Move our provided sample `open_loop.py` file to your desktop and then run the following commands:

```
roscd winter2024
mkdir scripts
cd scripts
mv ~/Desktop/open_loop.py ~/ros2_ws/src/winter2024/scripts
```

- Back to the terminal, you can run it in two ways. One is to feed this script as input to the Python program in Linux, as shown below.

```
python3 open_loop.py
```

- The other way is to run it as a regular executable in Linux. In this case, you need to first grant the execution permission to this Python script. This step only needs to be run once.

```
chmod +x open_loop.py
```

- Now you can see that this file is in green color when you `ls` the current directory in the terminal. This is how Linux terminal distinguishes executable (in green or highlight) and non-executable (in white).
- Then you can run it by command

```
./open_loop.py
```

- OK. Now please edit this open_loop.py file such that it can visit the four waypoints we mentioned before. Submit this script alongside a few screenshots once you are done.