

LABORATORY III

LIDAR Mapping with Wheel Odometry

PRA0102 - Group4

Gun-hui Han

Seonghak Lee

Jisoo Park

Part 1 : Vehicle Calibration

- **What path the robot was driven in each experiment.**

*** How did you drive the robot to determine the wheel radius? How much did it rotate? How far did you drive forward?**

The robot is driven in a straight line to determine the wheel radius. We records the number of wheel rotations as it moves forward. The average number of rotations is calculated to determine the wheel radius. The DRIVEN_DISTANCE defined as 0.75 meters, along with the number of wheel rotations, are used to calculate the wheel radius by using the following equation.

$$\text{radius} = \text{DRIVEN_DISTANCE} / (2 * \pi * \text{avg_rotations})$$

*** How did you drive the robot to determine the wheel separation? How much did you rotate? How far did you drive forward?**

The robot is rotated in a place to determine the wheel separation. The robot rotates a specified number of times in place. In this process, one wheel moves more than the other, causing the robot to rotate. The distance is measured to calculate the baseline. Instead of a forward distance, we use the distance each wheel moves during the rotation. The average total distance moved by the wheels is used to calculate the robot's baseline. The distance is determined by the wheel radius from previous part and the number of rotations.

- **How does your code work or should work? How are the parameters determined?**

l3_estimate_wheel_radius.py - This code calculates the wheel radius by driving the robot a specific distance and counting the encoder ticks. It uses the ticks to compute the average rotations of the wheels and applies the formula for the circumference of a circle to find the radius.

l2_estimate_wheel_baseline.py - This code estimates the distance between the wheels by rotating the robot in a place and measuring the encoder ticks during this rotation. It calculates the total distance each wheel travels and then calculates separation using the wheel radius.

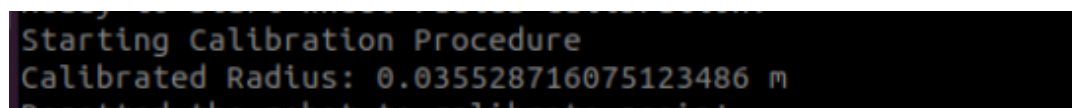
DRIVEN_DISTANCE is predetermined distance the robot travels straight forward.

TICKS_PER_ROTATION is encoder ticks in one full rotation of the wheel, also predefined.

WHEEL_RADIUS is determined from the robot's specifications or from the value obtained from l3_estimate_wheel_radius.py

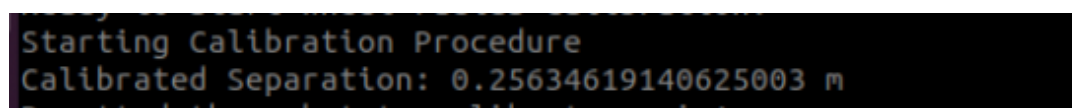
- **What values did you get for the wheel radius and baseline?**

Calibrated radius : about 0.034m



```
Starting Calibration Procedure
Calibrated Radius: 0.035528716075123486 m
Resetted the robot to calibrate again
```

Calibration baseline : about 0.256m



```
Starting Calibration Procedure
Calibrated Separation: 0.25634619140625003 m
Resetted the robot to calibrate again
```

- Does these values match those given in the data sheet? Identify one possible source of uncertainty or bias that made your answer differ from the factory calibration.

The calibrated values obtained are close to the real values from the data sheet, however they are not exactly the same.

*** Identify the source of uncertainty or bias.**

Slippage. When the robot moves or rotates, it is possible for the wheels to slip slightly, especially on smooth or uneven surfaces.

*** How does the source of uncertainty or bias affect your measurement?**

This could result in inaccurate encoder ticks, causing uncertainty in measurements.

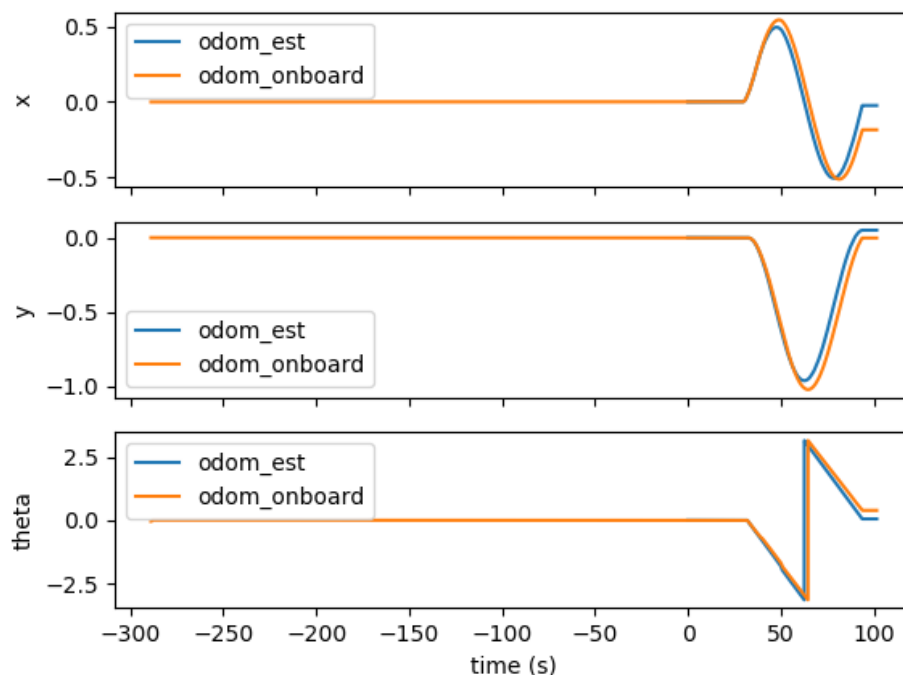
*** How could you mitigate this source of uncertainty or bias?**

Adjust the robot speed to move slower, as slower speeds can reduce slippage.

Part 2 : Motion Estimation

Two plots from the two experiments comparing your odom estimates with the onboard one.

There were slight discrepancies between odom estimates and onboard. Please note that the initial section of the graph, which is plotted flat with zero values and the x-axis time is negative, is due to a delay caused by the turtlebot_bringup functions. Please focus on the subsequent part of the figure.



- Briefly discuss the results

*** How is your estimation compared to the onboard one? Name one possible source of error that accounts for the differences.**

Sensor noise can introduce error to odometry estimation. Which leads to discrepancies when compared to the onboard measurements.

*** How are the estimates compared to the actual trajectory you observed? Name one possible source of error that accounts for the differences.**

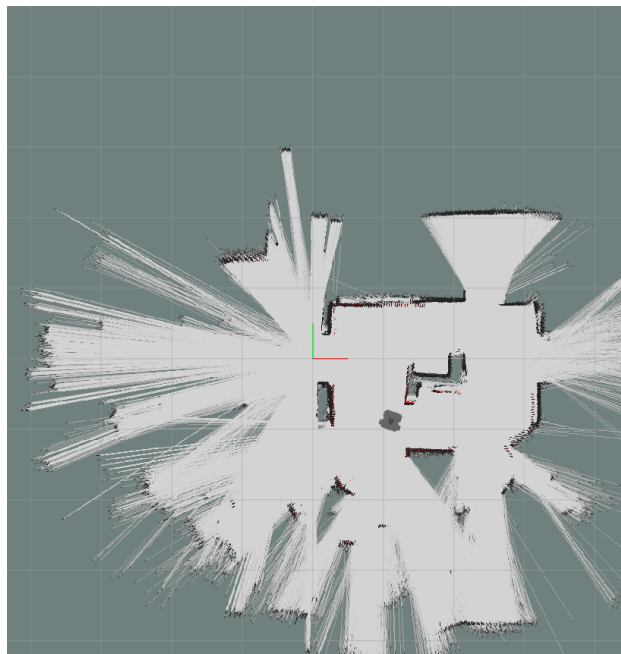
Wheel slippage can lead to discrepancies with the actual path.

Part 3 : Occupancy Grid Mapping

Include a picture of the mapped environment - similar to the video the mapped environment only needs to show the closed perimeter of the simulation environment.

- Describe how the sections you coded works or should work.

The code calculates and publishes robot's wheel odometry, comparing with onboard odometry using encoder data from the wheels. It calculates position and orientation changes from encoder ticks and updates robot's estimated position and velocity.

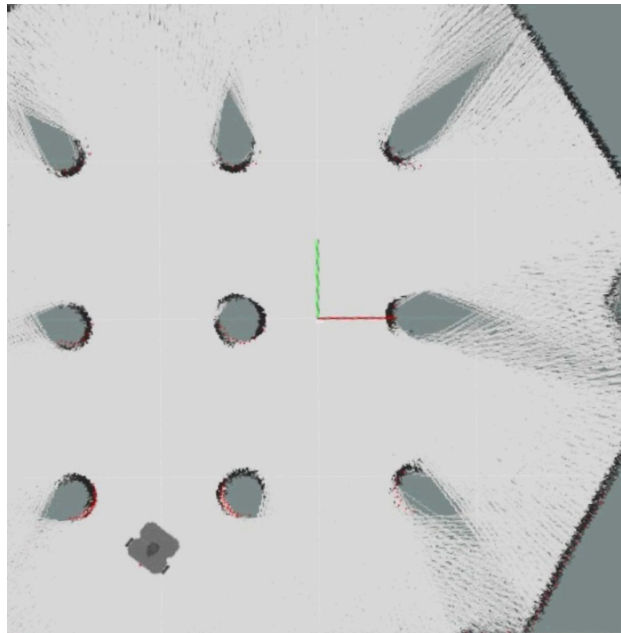


- Explain a potential source of error in this mapping algorithm.

The algorithm uses certain functions to change coordinates, like 'convert_pose_to_tf', 'euler_from_ros_quat', and 'ros_quat_from_euler'. If there are mistakes in how these transformations are made, or if there are problems with the math calculations, the map it creates might not be accurate.

Part 4 : Mapping on a Real Robot

– Include a picture of a map of the environment.



Demo marked by TA during lab session.

– Discuss the results. Explain a potential source of error that did not present in simulation.

When mapping with a real robot, it can be observed that mapping works similarly well as with occupancy grid maps done previously. However, When applying this algorithm in the real world, constraints must be considered. Issues such as slippage of the wheels can arise in actual conditions, leading to problems in accurately measuring the robot's movement or speed.