University of Maryland College Park

Professional Master of Engineering

Robotics Engineering

# ENPM809E - Python Application for Robotics - Final Project

***Students:***
Nupur Nimbekar (116894876)

***Lecturer:***
Zeid Kootbally

August 20, 2020

**Task**

The goal of this project is to solve a maze problem using only one Turtlebot 3 robot which comes equipped with a laser scanner.
We had to achieve the following tasks:

- Detect a wall
- Follow the wall
- Avoid collision with obstacles
- Find the exit

We had to accomplish the Tasks automatically.

**Summary**

1. **Approach for the Project: -**

   I have implemented a basic wall following algorithm which can follow a Right wall or Left wall depending on the surrounding.

   Data is extracted from LaserScan node in ROS and the data is stored in a numpy array. Later a dictionary was made to store values at certain angles to define left, front, and right side of the robot.

   Based on which side of the robot has more distance from obstacle, the bot adopts the algorithm for either Left wall follower or the Right Wall follower.

2. **Data collection from LaserScan node: -**

   The robot can access distance information for all 360 angles at every point. But we don't necessarily require all of these information. Hence, the first thing we did was to store the values at specific angles in a dictionary. this dictionary has left defined between nodes 0-40, right between 70-110, and right between 140-180. This was possible after storing data in some customized ways.

   Once we had the LaserScan data we also created a ROS subscriber node which subscribes to a callback function.

   We also create a Velocity Publisher node so that we could move the robot based on the algorithm explained in the next part.

3. **Algorithm for Wall Follower: -**

   The algorithm started after defining a global variable follow_dir which will be $-1$ till the wall is selected.

   We set a minimum ($a$) and maximum ($b$) threshold to define the boundaries in the algorithm.

If we reach a point where robot has wall to its front at distance which is less than maximum threshold, it will check the distance from obstacle of both the sides. If the left side is greater than the right side, robot will turn left hence adopting the right Wall follower algorithm. And if the robot turns right, it will adopt the left wall follower algorithm.

Depending on what algorithm is adopted, the state of the follow_dir variable will be set to either 0 or 1.

Inside the algorithm, as long as the wall is minimum distance way from the bot it will keep going straight. If it is a Right wall following robot and it reaches a corner where we have wall on the front and Right side, it will take a left turn. And if the robot reaches a corner in the map where the wall ends, based on how it maintains distance from the wall, it will go around the wall.

States are defined in the main function block to enable different action set for the robot. Hence it can take a Right turn, Left Turn, Go straight, or take diagonal turns on both the sides depending on the type of algorithm selected.

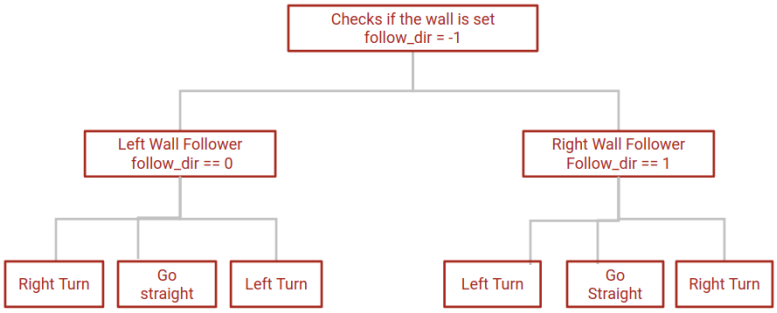
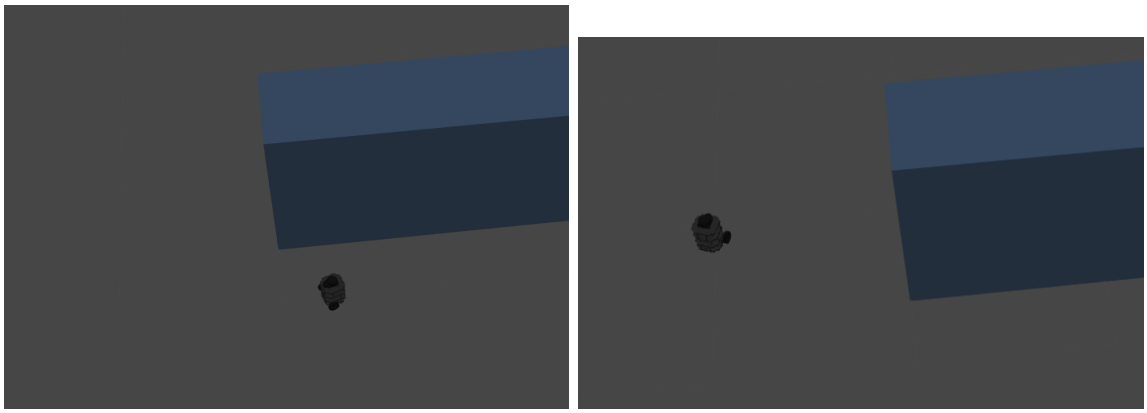
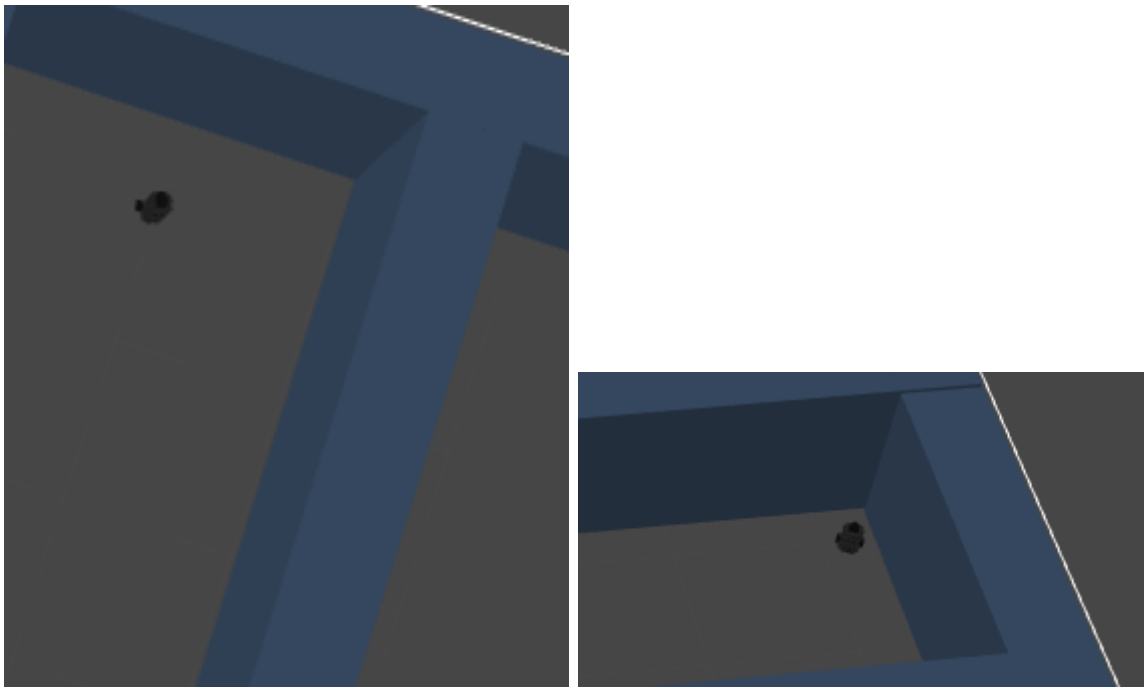



Figure 1: Diagram to show the conditioning used to determine the wall to be selected.



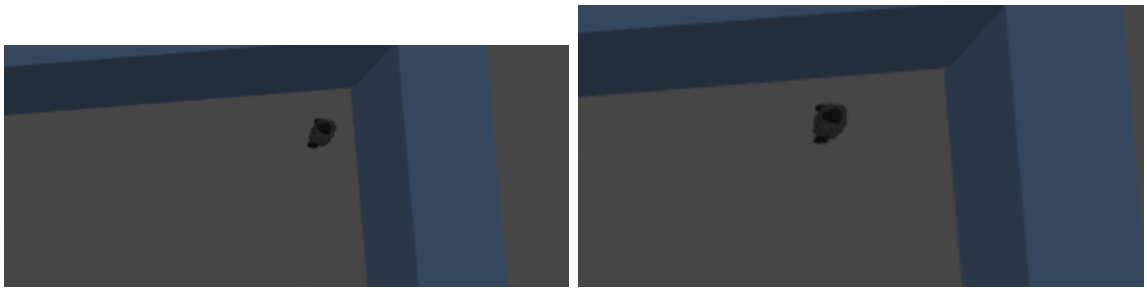
(a) Corner 1 (b) Turn Right action

Figure 2: Images from Simulation around outwards corner or end of a wall



(a) Corner 2 (b) Turn left action

Figure 3: Images from Simulation around Inwards corner



(a) Corner 3 (b) Turn left action

Figure 4: Images from Simulation around Inwards corner immediately after the first

**Problem Faced**

The main issue I initially faced was with implementing the action set to make the bot follow the wall. This I overcame by using a global variable to set a follower. This not only made the wall follower more functional but also provides with a more clear and versatile algorithm as both left and right hand rule can be used. This makes the automation of robot more flexible.

Second issue was mainly understand the importance of preprocessing the data that we receive. Once proper range for set for the sides, it made simplified the process of designing the algorithm.

Most important issue for me was to implement the wall follower without accessing the goal position. This was because due to misunderstanding the instructions as I was not aware if accessing the goal point was legal, but this provided me with more challenges and helped me understand how to develop a proper algorithm with minimum preset data.

With a proper python script to go to goal we could use this package to reach a specified goal point.

**Thank you for your guidance Professor**