Project 2 Deliverable 2

John RIpple
CSCI
Colorado School of Mines
Golden, United States
jripple@mines.edu

***Abstract*—This document discusses one method to create a wall following robot.**

I. Introduction

A common application for robots is object detection and avoidance. This paper discusses how a robot simulated in Gazebo and ROS is able to follow one wall using a Q-table for the state decision table.

II. Q-table

Q-table implementation for robotic control allows for extensibility into more states and the possibility of using machine learning to better optimize choosing states. The Q-table designed for this application was manually filled out in order to move from one state to the next with 100% certainty.

TABLE I
Definition of States

State	Front	Left	Right
0	>	>	>
1	<	>	>
2	>	>	<
3	>	<	>
4	<	>	<
5	<	<	>
6	<	<	<
7	>	<	<

A 360-degree liDar sensor is placed on the robot and data was read using the /scan topic [1]. The laser with the smallest value at +/- 10 of 0, 90, and 270 degrees are used to measure the distance in front, to the left, and to the right of the robot respectively. Each state is defined as the sensors reading distance values greater than or less than the tolerance [2]. The states in Table 1 were chosen because it covers each side of the robot with sensors while still having minimal states to reduce the curse of dimensionality.

The Q table shows that for every state there is only one action that should be taken with no probability involved. These actions were chosen because they were the minimal actions required to find and follow a wall. If a robot is following a wall it only needs to turn in one direction or use a P controller while following which is encapsulated in the three actions chosen in Table 2.

TABLE II
Q Table of States and Actions

States	Find wall	Turn Left	Follow Wall
0	1	0	0
1	0	1	0
2	0	0	1
3	1	0	0
4	0	1	0
5	0	1	0
6	0	1	0
7	1	0	0

III. Test Environment

The test environment was setup to be a maze that the robot could follow a wall around. The robot starts in the center and finds a wall then follows that wall around the course.

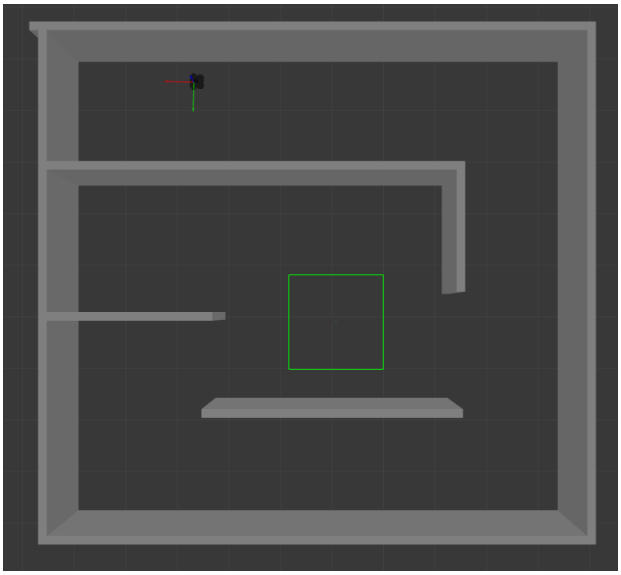

Fig. 1. Robot in test environment.

IV. Performance of the robot

Running the robot through the test environment allowed for a full circuit of the course. A class-based implementation with two P controllers were used to make the robot follow a wall [3]. The P controllers would rotate the robot while it was not at the desired distance from the wall based on the right sensor distance. Since a P controller was used, the robot does

wobble as it tries to get to the desired distance. Once the robot was sufficiently close to the desired distance a secondary P controller took effect that read the liDar at angles 300 and 240 and focused only on keeping those distance measurements equal allowing for the wobble of the first P controller to be negated. A PID controller was tested as well but did not yield better results than the two P controller method [4].

Since a Q table implementation was used with sensors on the left, front, and right of the robot, the code is generalizable to run a full course including turns.

To view a video of the robot demonstrating wall following capabilities click the link below.
Robot wall following link

REFERENCES

[1] "Sensor_msgs/laserscan message," sensor_msgs/LaserScan Documentation, Mar. 02, 2022. [Online]. Available: http://docs.ros.org/en/noetic/api/sensor_msgs/html/msg/LaserScan.html. [Accessed: Mar. 01, 2023].

[2] M. Arruda, "Exploring ROS with a 2 wheeled robot 7 - Wall Follower Algorithm," The Construct, May 23, 2019. https://www.theconstructsim.com/wall-follower-algorithm/ (accessed May 23, 2019).

[3] A. Parundekar, "turtlesim/Tutorials/Go to Goal - ROS Wiki," Ros.org, Oct. 07, 2021. http://wiki.ros.org/turtlesim/Tutorials/Go%20to%20Goal (accessed Mar. 01, 2023).

[4] M. Lundberg, "simple-pid: A simple, easy to use PID controller," PyPI, Apr. 11, 2021. https://pypi.org/project/simple-pid/ (accessed Mar. 02, 2023).