Michigan Technological University

MEEM 4707: Autonomous system

Spring, 2024

Lab - 4

By

Colton Kreicher and Amanda West

Example)

Name	Logic Build	Coding	Report Writing	Total
Amanda	60%	30%	60%	150%
Colton	40%	70%	40%	150%

Problem 1

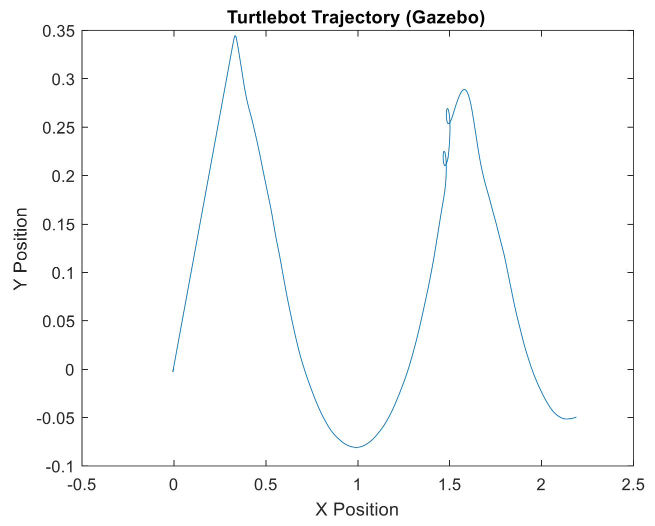

Figure 1: TurtleBot Trajectory in the Gazebo Enviornment

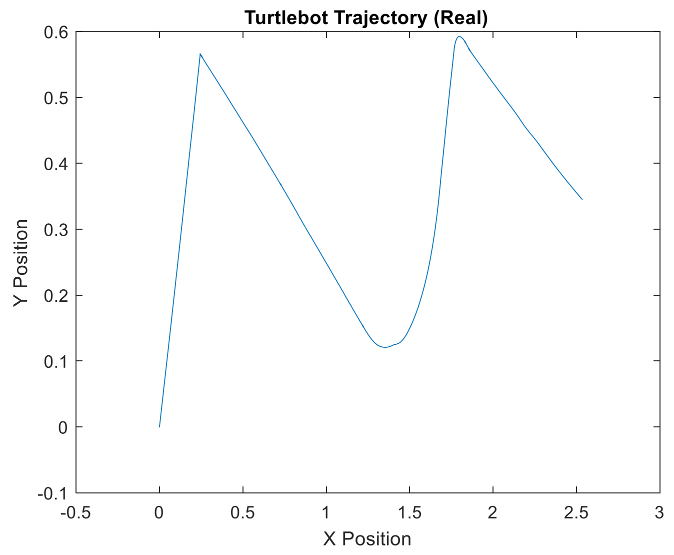

Figure 2: TurtleBot Trajectory in the Real-World Environment

Problem 2

Our approach was to use a state machine, where we switch between different states/modes sequentially as we move toward the path; the stages used were LOCATING, POSITIONING, ALIGNING, and FOLLOWING.

- After discarding the first few seconds of data to ignore bad LIDAR readings, the system starts in LOCATING mode.
- In LOCATING mode, the robot rotates CCW until it is within ±5 degrees of facing the closest point, and then switches to POSITIONING mode. This faces us toward the closest point on the wall.
- In POSITIONING mode, the robot moves forward at 0.1 m/s until it is within 0.5 m of the wall, stops, and then switches to ALIGNING mode. This places us next to the wall, facing it.

- In ALIGNING, the robot rotates CW until the closest point on the wall is within ±5 degrees of the left side of the robot, and then switches to FOLLOWING mode. This puts the wall precisely on our left side, pointing us towards the end of the wall.
- In FOLLOWING mode, the robot's velocity is set to a constant 0.1 m/s, and the robot's angular velocity is set to a constant kP times the angular error. This operates as a heading-based P-loop (PID with only kP), where the angular setpoint is 90 degrees (wall is kept precisely on the left side of the robot). The net result is that the robot attempts to move parallel to the wall, even as it zig-zags.

To further improve the performance of the model, a controller which accounts for the distance from the wall as it moves along it would be ideal. This could be accomplished by using trig to calculate the distance from the wall some distance ahead (ex: 0.2m), and controlling the angular velocity based on that feedback rather than the relative angle. If the time spent was warranted, however, the best model is likely the one which plots a path around the C-space initially observed, and then uses PID to regulate the angular speed of each wheel using velocities derived from the planned path as setpoints.

Objective

In this lab assignment, we were meant to alter a program that would allow the TurtleBot to find the nearest wall, move until it is 0.5 m away from the wall, and then follow it until it reaches the end, traversing around an inner and outer corner.

We test this program in the gazebo environment, collect the data and plot the trajectory, then do the same in the real-world environment.

Approach to achieve the Objective

Our approach was to use a state machine to move from locating the wall, to positioning ourselves at 0.5 m from the wall by moving forward at 0.1 m/s, to aligning the TurtleBot parallel to the wall, to finally following along the wall at a constant distance.

We integrated a PID controller into the code which was adjusted to allow the turtle bot to regulate its heading relative to the wall next to it.

Challenges faced and countermeasures taken

We encountered a few challenges during this lab. The first was in the gazebo environment, when we were turning to face the wall in our LOCATING mode, the angle reading would overshoot our tolerance, making the robot spin multiple times. We fixed this issue by increasing our tolerance from 1 degree to 5 degrees to ensure the values being recorded would pick up when the angle was close to 90 degrees.

The TurtleBot would also overshoot when it reached the first corner. The large change in distance values caused the TurtleBot to make a very wide turn. We fixed this problem by altering the proportional value on our PID controller and tuning the follow distance.

The final issue we encountered was when transferring the code from the gazebo environment to the real-world environment. We did not initially change the values to incorporate the real world TurtleBot, therefore, it would not follow the same path as the gazebo TurtleBot. We had to reupload the code after changing the scan array np.zeros.

The difference in strategy: Pre-lab vs. Lab strategy

The pre-lab had prepared us for understanding how the code would take in the positioning data and convert it to distance from the wall and the angle from the wall.

Using this knowledge, we were able to integrate the distance and angle readings to move to the wall from our initial position, and then along the wall at the desired distance and speed.

Observations and Learnings

Using a Proportional controller was an idea that we used so we had more control over tuning the TurtleBot's movement. This decision allowed us to regulate the error in positioning. When we noticed that the program was returning data at a slower rate than initially expected, we learned to adjust by increasing our tolerances.

We additionally learned that while the Lidar performs very well on the actual robot, providing a higher resolution than simulated, the physical drivetrain is far from reliable, even over short distances. Both the delayed response and asymmetrical performance indicated that future versions will need to treat open-loop segments with extreme caution.