Lab 3 Report: Safety Controller and Wall Follower

Team 12

Adan Abu Naaj
Amina Luvsanchultem
Andy Yu
Dylan Gaillard

6.4200: Robotics Science and Systems

March 15, 2025

1 Introduction

Author(s): Andy Yu

In this lab, we are tasked with implementing a **safety controller** and a **wall follower**. This is our first team-based lab deployed on hardware and represents a step toward building an autonomous system through a robot that can navigate autonomously, given reasonable conditions. One of our specifications is to prevent crashes using our safety controller while maintaining flexibility in our other tasks. Our other goal is to implement code for our robot car to reliably drive a fixed distance from a designated wall at a given speed.

The robot's LiDAR scanner is crucial in achieving these specifications. The scanner provides distance measurements to the nearest obstacles around the car, which we utilize in our **safety controller** and **wall follower** code.

Our **safety controller** implementation involves publishing an overriding stop command based on LiDAR data. A stop command is issued if our code determines that an obstacle is too close to the front of the car.

We take a case-based approach to our **wall follower**. The cases are determined by fixed detection windows, each with a trigger threshold. Our algorithm identifies the car's case based on which windows are or are not triggered. The algorithm then switches the distance measurement calculation method and LiDAR detection window according to the determined case. The calculated distance is then piped into our PD controller to determine the car's steering angle so it continues following the wall.

2 Technical Approach

Author(s): Dylan Gaillard

2.1 Problem Statement

The goal of this lab is to create working ROS2 wall follower and safety controller packages on the racecar robot. As parameters, the wall follower must accept a desired distance from the wall, side to follow, and velocity. The safety controller must be tuned to prevent collision with any obstacles in the immediate path of the car. It must also take priority over the wall follower. The average and variance of wall distance in multiple wall-following situations will later be evaluated to determine the implementation's effectiveness. Now, we describe our right wall-following implementation. Note that it generalizes to a left wall as well; when there is a significant difference in method between the two sides, it will be noted.

2.2 Initial Setup

Our initial setup for this lab consisted of a racecar robot equipped with various sensors and:

1. an onboard LiDAR sensor
2. 4-wheel-drive and steering motors (an NiMH battery)
3. an NVIDIA Jetson Nanodeveloper kit (and battery) running ROS2 in a docker container

4. a router equipped with ethernet for communicating with the Jetson
5. a Logitech joystick controller for teleoperation

2.3 Wall Follower

Our technical approach to the wall follower separates its expected behavior into three categories: Outer, Inner and Straight wall following, with outer and inner corners being defined by turns towards/away from the wall, respectively. These categories inform our LiDAR data processing, robot control defined in the following subsections.

2.3.1 LiDAR Window Slicing

Subscribing to the topic `/scan`, we capture all of the car's current LiDAR measurements as `LaserScan` data, from which we extract range and angle measurements. These are defined such that $\theta = 0$ points directly ahead of the car, with positive counterclockwise. To ensure we use only points relevant to wall following, we mask all data to a range of 7 meters. Our LiDAR data callback function processes the data as follows. Note LiDAR data slices, expressed as $[\theta_1, \theta_2]$ should be negated and reversed for the left wall case ($[-\theta_2, -\theta_1]$).

Detection Windows: We define a two detection booleans `front_close` and `side_close`, related to the `front` and `side` windows as defined below.

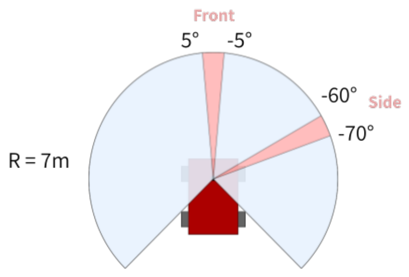


Figure 1: Detection windows for the two detection booleans.

These windows measure mean distance within slices of the LiDAR data within predefined angle ranges. If the mean distance measurement falls below the window's assigned threshold, we store this as `True` (close), and `False` (far) otherwise. The `front` and `side` windows contain all (masked) data in the angle range $[-5, 5]$, and $[-70, -60]$ degrees, respectively. These booleans allow us to detect Inner/Outer corners, and Straight walls in the following order of priority:

1. Outer Corner: Side wall far (`side_close` = `False`)
2. Inner Corner: Front wall close, side wall close (`front_close` = `True`, `side_close` = `True`)
3. Straight Wall: Side wall close (`side_close` = `True`)

We check these cases in the LiDAR callback function to determine the wall following behavior given current measurements.

Wall Slices: In addition, we define two slices of LiDAR measurements to be used for later interpolation, defined below.

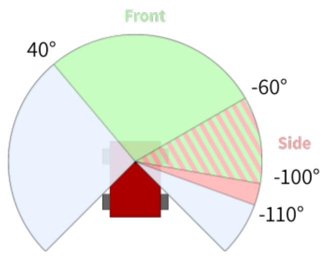


Figure 2: LiDAR slices for interpolation. The slices used are determined by cases (elaborated below).

We use `front` and `side` windows with ranges $[-100, 40]$ and $[-110, -60]$ degrees, respectively. When in the Straight or Outer Corner cases, we use the `side` window. Otherwise, we use the `front` window.

Initially, we only used a front detection window, but this proved insufficient for Outer corners, as our implementation could not distinguish between "T-shaped" junctions (Outer corners) and standard Inner corners, as interpolated data (described later) was relatively similar in both cases.

2.3.2 LiDAR Interpolation

Given sliced (and masked) LiDAR measurements, we use linear interpolation to determine the shape/slope of the wall.

Given any LiDAR (d, θ) measurement of range and angle, respectively, we can find its (x, y) position relative to the car using the formulas

$$x = d \cos\theta$$

$$y = d \sin\theta$$

Using the built-in `numpy` method `polyfit`, we interpolate a line of the form $y = mx + b$, in the coordinate system defined with the LiDAR sensor at the origin. This provides a linear approximation of the surface being followed, as shown:

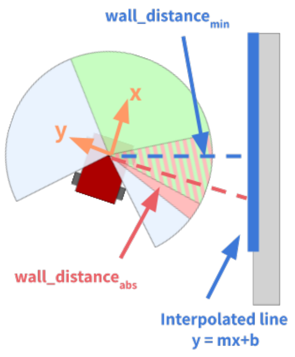


Figure 3: Visualization of wall interpolation and the distance calculations we gather.

Using this line, we can find two distinct measurements of wall distance, minimum (wall-perpendicular) and absolute (rover-perpendicular) distance, defined by the equations below.

$$\texttt{wall\_distance}_{\texttt{abs}} = |b|$$

$$\texttt{wall\_distance}_{\texttt{min}} = \frac{|b|}{\sqrt{1+m^2}}$$

In the Outer Corner case, we use absolute distance. In all other situations, minimum distance is used. We can then feed these wall measurements to the PD controller defined in the following section.

While minimum distance provides an accurate measurement of wall distance given any orientation of the racecar, we use absolute distance in Outer corners to address the distinct behavior of the Outer case's interpolation. Since most LiDAR points in Outer corners are located after the turn, the interpolated wall is roughly perpendicular to the racecar and minimum distance underestimates the true wall distance, and PD control (described later) turns the car away from the wall. Absolute distance overestimates, but PD correctly steers towards the wall.

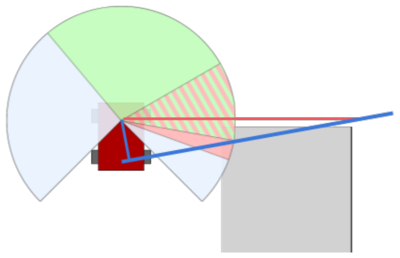


Figure 4: The robot turns more around corners due to the absolute distance to the wall being greater than minimum distance.

2.3.3 PD Controller

To control the car, we use a standard Proportional-Derivative (PD) controller formulation. The current error is defined as

$$e_t = r - d$$

where d is the current distance of the vehicle from the wall, and r is the reference (desired) distance of the vehicle from the wall.

The appropriate steering angle u is calculated as

$$u = K_p * e_t + K_d * (e_t - e_{t-1})$$

.
where e_{t-1} is the error from the previous timestep, and $(K_p, K_d) = (1.7, 0.2)$ are tunable parameters. This steering command is then published on the topic `/vesc/high_level/input/nav_0`.

Solely proportional gain yielded an oscillatory response, so derivative gain was included as well. This provides a satisfactory response, so no integral term was included.

2.3.4 Safety Controller

We also defined a safety controller node/package to issue an overriding stop command if the racecar is on a collision trajectory.

2.4 LiDAR Data

Reading the aforementioned `/scan` topic, our safety controller issues an overriding stop command if any range within the slice $[-8, 8]$ degrees falls below the threshold distance:

$$\texttt{stopping\_distance} = \texttt{STOP\_COEFFICIENT} \times \texttt{self.VELOCITY}$$

Traveling 1 meter per second, the stop threshold is 0.6 meters, for example. This ensures the safety controller does not underestimate the distance required to stop the rover at higher velocities. This formula was determined appropriate by empirically testing different coefficient values and assessing final front distance in collision scenarios.

2.4.1 Mux Structure

The safety controller node subscribes to the mux (topic)

`vesc/high_level/ackermann_cmd`

to determine the current velocity at which the vehicle is traveling by reading the published `AckermannDriveStamped` messages. The current velocity is then set as `self.VELOCITY` to be used in calculating the threshold `stopping_distance`. Once within the stopping distance threshold, the safety controller node publishes a zero-velocity `AckermannDriveStamped` command to the low-level mux (topic)

`vesc/low_level/input/safety`

overriding all wall follower commands.

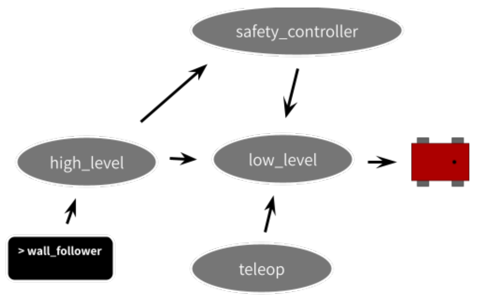


Figure 5: High level diagram of our muxes. Wall follower publishes to the high_level mux. Safety controller reads from the high_level mux and publishes to the low_level to override the high_level mux if necessary.

3 Experimental Evaluation

Author(s): Adan Abu Naaj, Amina Luvsanchultem

Our experimental evaluation aimed to test the functionality of the safety controller and wall-following module. We have conducted qualitative and quantitative tests to assess the robot's behaviour in real-world scenarios and to evaluate the robot's accuracy by measuring the robot's distance from walls and obstacles.

3.1 Testing Approach

We designed a series of tests to cover key functional requirements defined in our technical approach:

- Maintaining a consistent distance from walls.
- Navigating inner and outer corners.
- Managing convex and concave wall shapes.
- Stopping when encountering obstacles at a certain distance from the robot's front.

To quantify each test case, we chose the following metrics:

3.1.1 Obstacle Detection

We tested the robot's safety controller to assess if the robot stops when presented with obstacles and measured the distance between the object and the robot to ensure it is within desired distance.

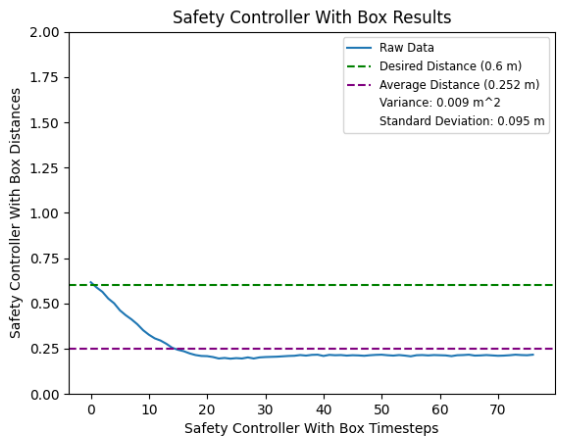
3.1.2 Distance from Wall

We measured LiDAR data of the robot distance from the wall it is following. We computed the average, variance, and standard deviation of this distance and plot it over the time steps to assess the accuracy and consistency of the robot in maintaining the desired 1-meter distance from the wall.

3.2 Test Cases

3.2.1 Safety Controller Performance:

We tested the safety controller by placing objects in the robot's path. The controller successfully stopped the robot upon detecting obstacles. Two scenarios were evaluated: a person walking to simulate pedestrian interactions and a box thrown in front of the robot. Given the data in Figure 6, we see that the robot successfully initiates the safety controller at a distance of 0.6 meters away from the obstacle or obstruction. The average stopping distances were 0.483 m from the person and 0.252 m from the box, indicating timely responses. The average distance is smaller for the faster objects (e.g. box) due sudden actions.



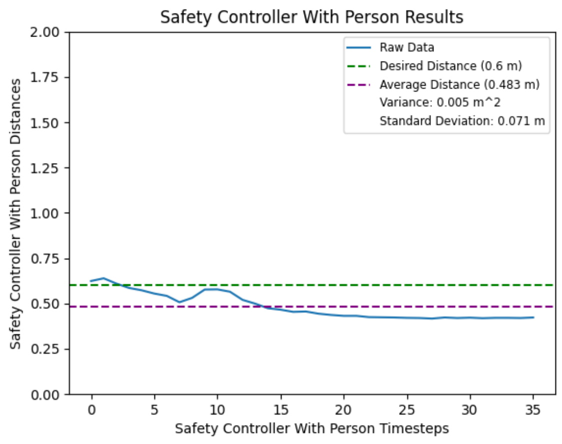
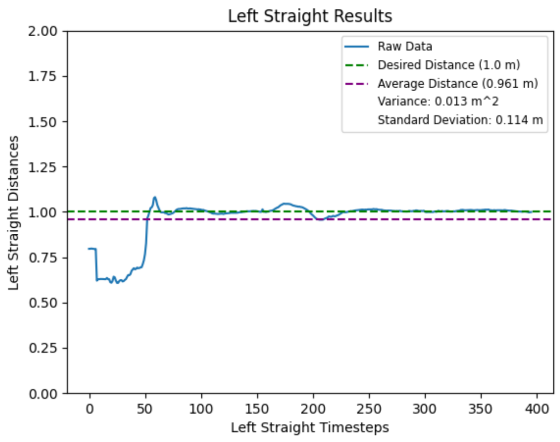


Figure 6: Safety controller distance data shows that the robot stops effectively to prevent collision with an obstacle. More sudden obstacles result in a smaller stopping distance due to the robot's reaction time.

3.2.2 Following Straight Walls:

We tested the wall follower when the robot follows a straight wall on its left or right side. We observe whether the robot maintains a target 1-meter distance from straight walls on both left and right sides. Results demonstrate consistent behavior, with average distances of 0.961 m and 1.023 m for the left and right walls, respectively. A variance of 0.013 m^2 and 0.003 m^2 for the left and right walls suggests a low variability.



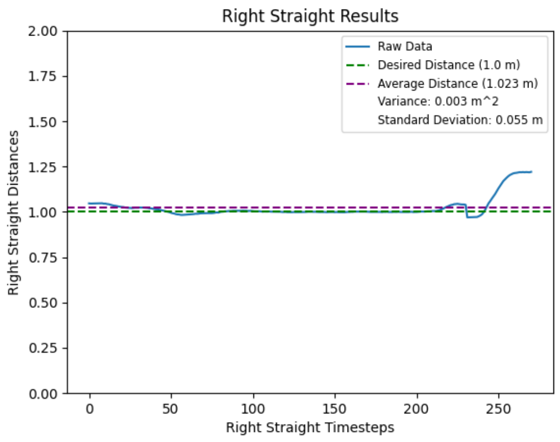


Figure 7: Left and right distance data for straight walls where there is minimal deviation from the desired distance. The robot maintains a consistent distance from the wall for both the left and right walls.

3.2.3 Inner Corner Performance:

We further tested the robot's ability to navigate sharp turns. The robot's performance on inner corners showed average distances of 0.93 m for left-wall following and 0.915 m for right-wall following, demonstrating accurate corner detection and appropriate steering responses. Although we notice deviations from the desired distance, the variance of the distance to the left wall is 0.037 m^2 and 0.047 m^2 for the right wall, meaning we maintained a low variability of the data.

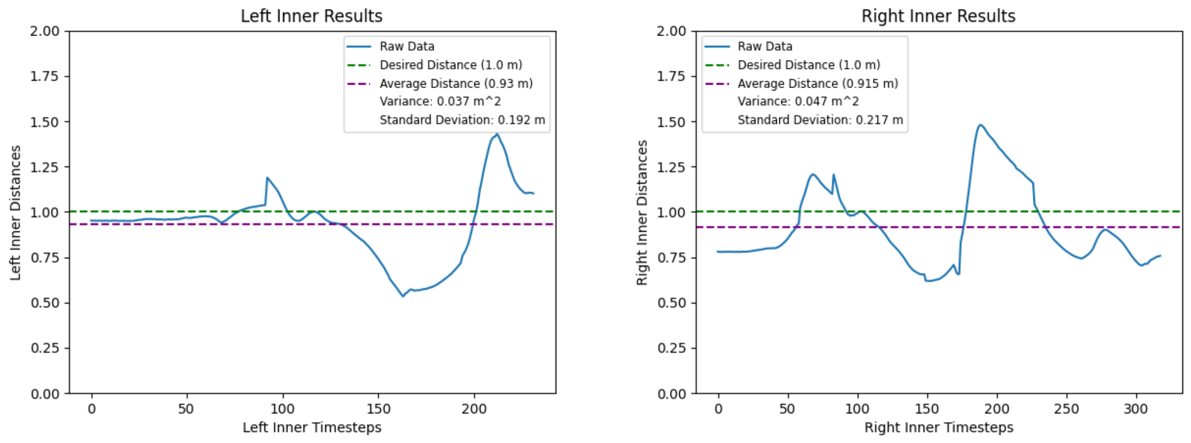
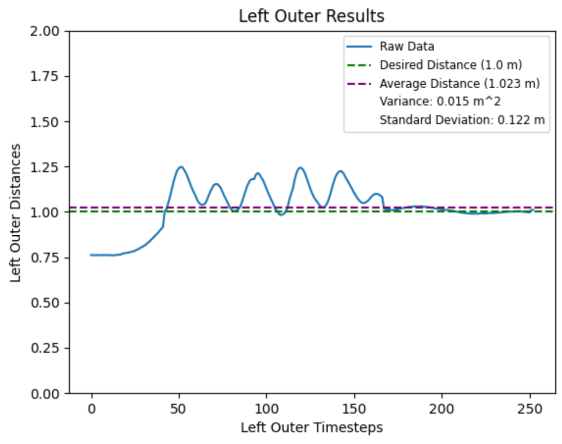


Figure 8: Left and right inner corner distance data shows that although there were a few deviations from the average distance, we still maintain an acceptable distance from the wall.

3.2.4 Outer Corner Performance:

For outer corners, average distances were 1.023 m from left walls and 1.036 m from right walls, confirming the controller's capability to handle outward-facing turns reliably. We noted slight deviations from the desired distance, reflecting the inherent complexity of managing wider-angle turns. However, these deviations remained within acceptable bounds. To show this, we see that the variance for the left wall was 0.015 m^2 and 0.003 m^2 for the right wall, suggesting a low variability of the data from the average distance to the wall.



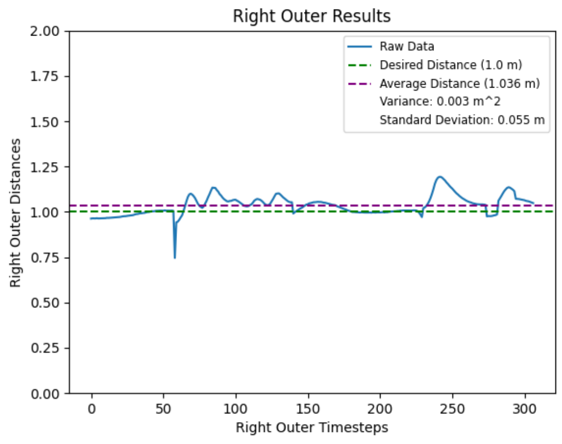


Figure 9: Left and right outer corner distance data plots show reliable control with slight deviations from the average distance within acceptable bounds.

3.2.5 Concave and Convex Wall Following:

Concave and convex wall scenarios tested the robot's adaptability to realistic wall shapes that oftentimes have small turns, gaps, obstacles, and uneven walls. For concave walls, the robot maintained an average distance of 0.968 m from the left wall and 0.989 m from the right wall. Convex walls yielded average distances of 0.929 m from the left wall and 0.974 m from the right wall. The data showcases consistent performance even in varying wall geometries. Given that the standard deviation is below 0.5 m for both left and right walls for the concave and convex test cases, the data demonstrates low variability overall.

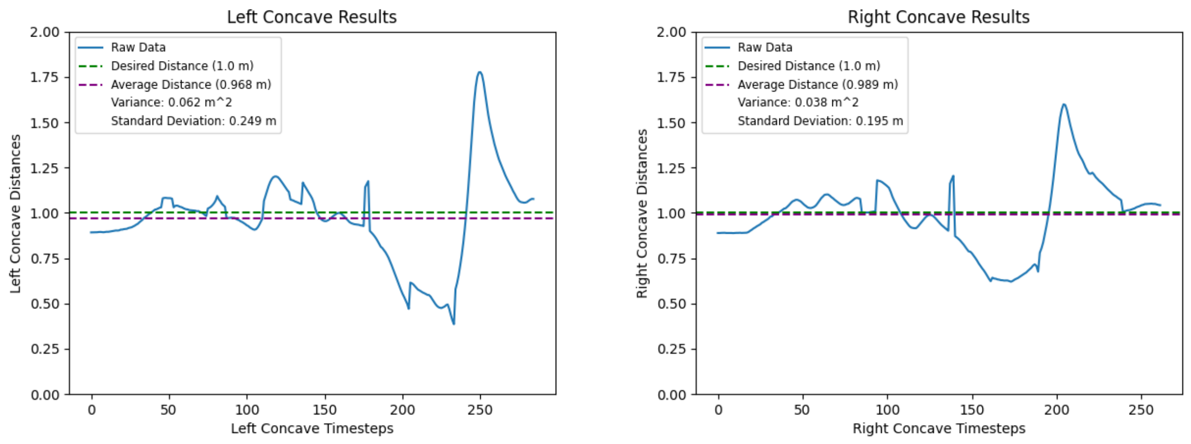


Figure 10: Left and right concave wall distance data shows an acceptable average maintained distance from the wall with some deviations in the data.

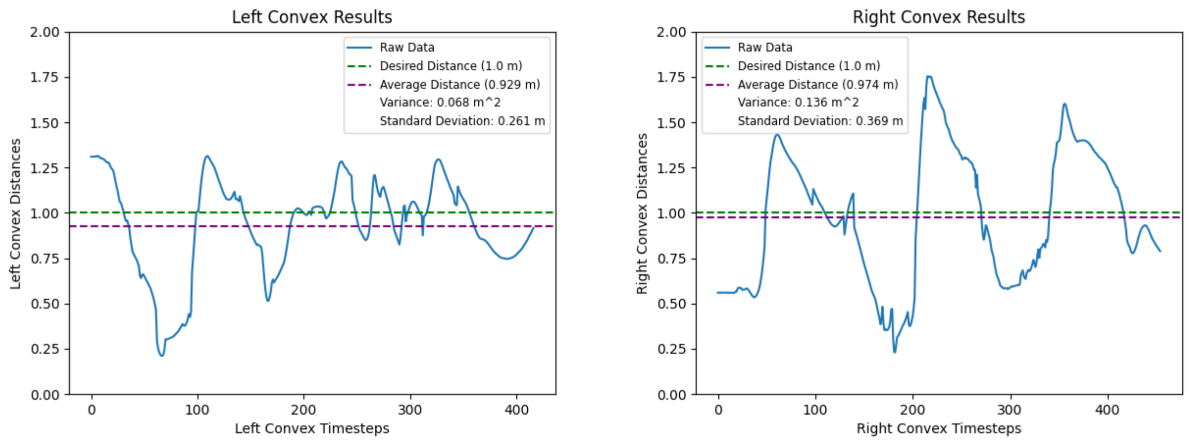


Figure 11: Left and right convex wall distance data shows some deviations in the data due to the nature of the walls we tested on, but the robot maintains an acceptable average distance from the wall.

4 Conclusion

Author(s): Amina Luvsanchultem

Our implementation of the wall-follower robot met our main objectives of this lab. The safety controller consistently prevents collisions by stopping within 0.6m of an obstacle while the wall-follower performed consistently in various scenarios with occasional deviations, maintaining an average distance close to the target 1 m away from the wall throughout our test cases. The results from our experimental evaluation, specifically in the cases of more complex geometry, demonstrate the robot's robust performance with minimal deviation from the desired distance. These results help to validate our case-based approach in that employing different detection windows based on the robot's position on the wall facilitates smoother navigation along complex wall geometries. The minor deviations that we noticed were related to navigating inner and outer corners where sharper corners presented challenges. However, these deviations did not significantly impact the performance and remained within an acceptable range from the target distance.

Future improvements include testing and tuning the wall-follower on higher velocities to ensure greater versatility of our wall-follower. Similarly, evaluating the safety controller at different speeds can enhance its adaptability and functionality in different scenarios. In conclusion, we successfully achieved reliable performance on both the wall-follower and safety controller in structured environments using our case-based PD controller implementation.

5 Lessons Learned

As a group, we gained greater insight into PD tuning protocol to produce a stable system with minimal oscillations. We also learned about managing ROS topic priorities in tmux, which allowed us to implement the safety controller in such a way that it would override our autonomous code in the event that the safety stop would need to be called.

5.1 Adan Abu Naaj

I learned lots of valuable skills from this lab. It was my first time deploying software on a hardware robot, and I gained valuable experience working with ROS nodes and topics. Running the nodes and seeing them operate on the robot helped me understand the practical side of software deployment.

Additionally, it was insightful to hear how my teammates approached the wall follower implementation after we each worked on it individually. This was my first experience using a PID controller, and despite not taking a controls class before, I gained a solid understanding of how PID works and how to apply it effectively.
This lab also served as an excellent introduction to autonomous vehicles, giving me a strong starting point in this field. I particularly appreciated the communication aspects of the lab, including the briefing and presentation. The presentation was highly professional, and I learned how to convey academic concepts both orally and in a formal written report.
Another aspect of this lab that I really enjoyed was designing both qualitative and quantitative test cases. At MIT, we are often provided with test cases and primarily focus on the implementation. However, this class offered a new perspective, more aligned with research and industry, where you have to be creative in figuring out how to evaluate your code effectively. This experience gave me valuable insight into the testing process and improved my problem-solving skills.
I'm incredibly grateful for my team. We had strong communication throughout the project, and I learned a lot from my teammates' insights and approaches. Overall, this lab was a rewarding experience that strengthened both my technical and teamwork skills.

5.2 Amina Luvsanchultem

My main takeaways from this lab focused on my technical growth from this lab in working with ROS nodes and topics, understanding the hierarchy of the ROS topics using tmux, communicating effectively with my group under a deadline, and problem solving while tuning parameters of the robot. Although, I had prior experience working with PID controllers from a controls class and through the NEET program, I felt that seeing it in action through the context of a wall-following robot, which mimics the concept of programming a real autonomous car, places the concept of PID control in a completely new light. I felt that I was able to better understand the control system as a whole, learn more about the time and effort that a well-tuned robot would require, and shift my perspective to the idea of programming an autonomous vehicle in the real world (e.g. implementing safety controllers). As a group we were able to bounce ideas off of each other and create an environment that I felt was engaging, efficient, and growth-oriented. The communication aspect of this lab provided me with a new perspective on technical presentations and engaging an audience during a presentation. The guidelines for the presentation focused on reducing the text on the slides and utilizing diagrams throughout the presentation, which was fairly different from the presentations that I had given in other classes. When creating and giving the presentation, we focused heavily on making the presentation seamless and easy to follow for the audience — a consideration that provided me with a greater awareness and appreciation for the aspects of presentations that make them engaging (e.g. animations, minimal text, etc). I felt that with the amount of time that we had spent working on the robot and creating the presentation allowed for us to create a deeper understanding of our robot and speak with greater confidence and understanding of the lab. Overall, I felt that this lab provided a valuable collaborative and technical experience, especially in being able to work with the real-world data from the LiDAR sensor and adjust our program to successfully maneuver the robot.

5.3 Andy Yu

The main lesson I learned was how little time we had for these labs. We spent much time making the robot follow the wall at a specific speed and distance, which meant our final result was not as robust as I would have liked. Furthermore, we spent considerable time debugging our robot's behavior around the blue pillar outside the lab classroom. Nonetheless, what resulted from the overspecific debugging was a complete overhaul of our wall following logic, which I am incredibly proud of. Despite not being tuned for all speeds and distances, I believe that our logic provides a fairly robust wall follower that can be generalized if the thresholds and variables are parameterized. Still, our time investments meant that relatively little time was left for quantitative evaluations. This is something I've traditionally struggled with, and it was made apparent through Professor Carlone's comments during our lab briefing. In regards to our briefing, this was the first time I had to present my work in such a professional environment. The Schwarzman Building, the glass conference room, and Professor Carlone's intimidating Italian eyes were daunting. I hope to be more comfortable in the future. This lab was also the first time I used ROS2 after hearing about it for a large portion of my time at MIT. Therefore, I learned a lot about subscribers, publishers, topics, nodes, and syntax. Furthermore, this was my first time implementing a PD controller after hearing about it so often. Overall, it was rewarding to implement what my MASLAB (IAP robotics class) teammates handled for me before.

5.4 Dylan Gaillard

My main takeaway from this lab is a better understanding of the overall software implementation and hardware testing loop that generally underlies work in robotics. Although the process was far more extended than I would have desired, I believe that working through it provided greater context in navigating how to approach the various deliverables associated with each lab in this class. Ultimately, this will allow our group to develop a more efficient workflow and general approach to completing work in this class. I also found that this lab provided a valuable opportunity to test and implement the theory of concepts that I had previously been exposed to on real hardware, such as a PID controller. Regarding communication, I was relatively comfortable before the lab in presentation/briefing settings, but after leveraging the various resources provided by the CI-M staff I feel that I have significantly refined my view of what defines a "good" presentation, subjectively and objectively. Moving towards our next labs, this is sure to improve how effectively I and my group communicate our ideas. Lastly, in terms of collaboration, I have developed a greater appreciation of the consensus decision-making process. I feel the various communication-intensive exercises prior to the lab provided some understanding, but relying on the method to approach a technical problem revealed nuances to the process that I would not have discovered otherwise. For example, better understanding exactly how strong consensus can be reached over multiple iterations of software, as compared to simply accepting a single solution, is something I gained from this lab.