Lab #3 Report: Wall-Following on the Racecar

Team #15

Lauren Carethers
An Bo Chen
Brian Li
Claire Lu
Rachel Lu

RSS

March 11, 2023

1 Introduction

Author(s): Lauren

The purpose of this Lab is to successfully implement a wall follower algorithm and safety control algorithm onto the racecar such that the car can a) successfully use LiDAR data to detect walls, follow a wall while maintaining a specified distance, and handle corners and curves and b) avoid collisions with obstacles, even when the car is moving at high speeds.

The racecar is fit with a LiDAR scanner that can measure the distances between the racecar and nearby objects. Generally speaking, the **wall follower** algorithm receives, slices, and filters this data based on the location of the nearest obstacle, the wall the racecar is following, and the desired distance that we want to maintain from the wall. This data is then used to find the closest distance between the car and the nearest obstacle and then inputted into a PD controller to determine the output steering angle. Along with the speed, the steering angle is then published as a drive command to the racecar so that the racecar can avoid obstacles while following the wall.

When the racecar gets too close to a nearby obstacle (i.e. significantly less than the desired distance away from it) our **safety control** algorithm comes into play. This algorithm works by taking in the distance data from the LiDAR scanner, determining if the distance between the car and the nearest obstacle is significantly small, and then stopping the car in its tracks if it is about to

collide with the obstacle.

Using these algorithms, our team set out to accomplish the following steps.

1. Drive the racecar in teleop mode and successfully visualize the LiDAR data.
2. Autonomously drive the racecar with our wall following code.
3. Prevent crashes using our safety controller while maintaining flexibility.

Given that the racecar is an expensive robot, the implementation of the above algorithms will ensure that the car won't harm itself beyond repair and thus will remain usable for future classes.

2 Technical Approach

2.1 Initial Set-up

Author(s): Lauren

Prior to this lab, our team's members each individually created a wall follower algorithm for the racecar simulation. Each of our algorithms split the LiDAR distances data into a variation of subsets, used either linear regression or another filtering method for finding the distances between the car and the nearest objects, and implemented some degree of control for the car's steering. Once the team was formed, one of our earliest tasks was to figure out who's wall follower code should be pushed to our robot. To determine this, each team member briefly described how their wall follower was structured and what score they received for the wall follower lab gradescope submission. The wall followers with high scores (9.0+) were first filtered, and then each member's code was reviewed for reliability and logic. Ultimately, the team chose to write a new wall follower that combined pieces of logic from multiple team members' wall follower algorithms. Our safety controller, however, was written from scratch.

2.2 Wall Follower

Author(s): Rachel (Lidar Data Slicing), An Bo (Distance Calculation), Lauren (PD Feedback Control), Claire (Corners Navigation)

To achieve a successful and efficient wall follower, our racecar needs to analyze and use the LiDAR data. It is crucial to translate the raw data from the LiDAR into useful pieces of information that we can incorporate into our program and use to tune our wall follower program. This results in a process of four main components.

1. LiDAR Data Slicing

2. Distance Calculation
3. PD Feedback Control
4. Corner Navigation

2.2.1 LiDAR Data Slicing

The first step towards achieving an efficient wall follower program is to analyze the data we got from the LiDAR. LiDAR data comes in the form of an array where each element in the array is the distance the racecar is from an object in the environment at a certain angle. Our LiDAR can measure distances at an angle between -135° and 135° with an increment of about 0.27°. This results in our LiDAR data having 1000 elements per callback. We divided this data into three sections: left, right, and front. This division is shown in Fig. 1. The left and right sections are used when the racecar is either following the left or right wall respectively. The front section is used for both corner detection (2.2.4) and the safety controller (2.3).

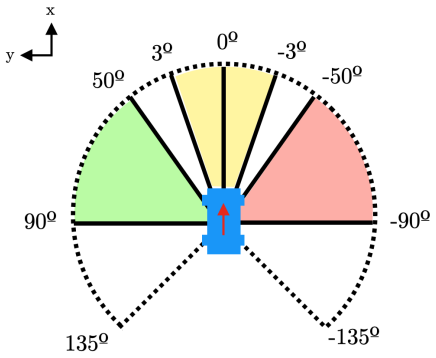

Figure 1: Sliced ranges of the racecar's LiDAR data where the entire region is -135° to 135°. Green represented the left partition, yellow represents the front partition, and red represents the right partition.

2.2.2 Distance Calculation

Once we sliced the LiDAR scan ranges and their corresponding angles into desired partitions, we filtered the ranges and angle partitions. First, we found the closest distance using the closest range data point to the racecar in range partition and then filtered the rest of the range and angle data points checking whether they were within MIN_DISTANCE $\pm$ DESIRED_DISTANCE $\cdot$ FILTER_FACTOR. This way we would reduce the amount of outliers that the racecar would use to

calculate the linear regression line to represent the wall that it is supposed to follow. The filter factor is determined through a tuning process where we analyzed how the resulting line is drawn in different environments. Then we calculated corresponding x and y coordinates using the filtered ranges and angles, where each x coordinates is the corresponding $\text{RANGE} \cdot \cos(\text{ANGLE})$ and likewise each y coordinates is the corresponding $\text{RANGE} \cdot \sin(\text{ANGLE})$. We used the *numpy polyfit* function to calculate a resulting slope and y-intercept. Finally, for our distance calculation we used Eqn. 1 to find the distance of a line in $y = mx + b$ from the origin:

$$\text{DISTANCE} = \frac{|\text{Y\_INTERCEPT}|}{\sqrt{\text{SLOPE}^2 + 1}} \tag{1}$$

2.2.3 PD Feedback Control

Our team implemented a variation of a proportional-derivative (PID) feedback controller into our robot so that it could handle both straight, uniform and curvy, non-uniform wall contours and corners. Simply speaking, a PID controller is a feedback loop that calculates the error between a desired setpoint and a measured processed variable and uses proportional (K_p), integral (K_i) and derivative (K_d) terms to apply a correction. In our case, the setpoint is our **desired** distance from the wall, and our measure processed variable is the car's **actual** distance from the wall. For our wall follower, we decided to implement a PD controller that solely used the P and D constants, K_p and K_d respectively, for simplicity. Using a purely proportional controller was not sufficient to eliminate oscillations and minimize error. We also found that our robot performed to our desired standards without the addition of an integral term.

Our PD controller parallels Eqn. 2 to give us our steering angle. $\frac{de(t)}{dt}$, which is determined via Eqn. 3, is dependent on

1. $e(t)$, our error and the difference between the car's **desired** distance from the wall and the car's **actual** distance from the wall
2. $e(t)_{prev}$, the previously calculated error
3. Δt, the time since the car received the last laser scan

$$u(t) = K_p \cdot e(t) + K_d \cdot \frac{de(t)}{dt} \tag{2}$$

$$\frac{de(t)}{dt} = \frac{e(t) - e(t)_{prev}}{\Delta t} \tag{3}$$

We determined the constant Δt to be 0.05 by taking the multiplicative inverse of the rospy rate. To find the constants K_p and K_d, we used a tuning process. First, we doubled K_p until the racecar started oscillating side to side, and then

we increased K_d until the car no longer oscillated. This process featured a lot of trial and error, but we eventually found that a K_p of 2.25 and a K_d of 0.5 allowed our robot to steer and turn effectively. The observations and tuning process can be seen in Fig. 6.

2.2.4 Corner Navigation

In addition to following a straight wall, we wanted our robot to be able to turn with the wall as it encounters both open and closed corners. We define an open corner as a corner where the wall curves away from the robot, and a closed corner where the wall curves in front of the robot, as seen in Fig. 2. We were able to achieve reliable turning on both open and closed corners by tuning the ranges of our LiDAR slices, and redefining the wall to follow when encountering closed corners.

Open Corners
To navigate open corners, we rely on our wall-following algorithm to properly turn the robot so that it continues to follow the wall. We had to experiment with the angle ranges that we use to slice the LiDAR data as well as our PD gains to achieve robust open corner turning that did not negatively impact our wall-following. Specifically, we found that including more data from the side and behind our robot helped with detecting when the wall bends away from the robot.

Closed Corners
To detect closed corners, we isolated a specific range of points from the front, spanning -0.1 to +0.1 radians. These points are used to calculate a specific front_error by filtering and drawing a line across points in the front, and finding the distance from the robot to that front line. The desired distance for this front_error is our desired wall-following distance times a front_factor parameter. We experimented with different values for front_factor, but found that front_factor = 4.0 achieved appropriate turning for the corners. With lower front_factor values, we detected closed corners too late and were not able to start turning in time.

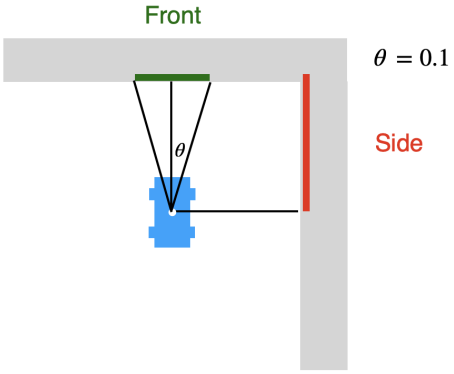

Figure 2: Separate front and side ranges of points, where the front range spans -0.1 to 0.1 radians. The front range is used to detect closed corners.

Once we have detected a closed corner, we redefine the wall that the robot is following. If we find that front_error is positive - that is, the robot is closer than (front_factor) · (desired_distance) to points in the range of -0.1 to +0.1 radians - we redefine the wall that the robot is following. We concatenate the points in the front with the points used to form the wall on the side, fit a line to those points, and define a new wall based on the combined front and side, as seen in Figure 3. This process creates a wall that cuts across the corner, and the robot is now following this artificial wall. From the robot point of view, the wall that it is following now suddenly curves in front instead of having a sharp, discrete corner, and we were able to successfully turn in closed corners.

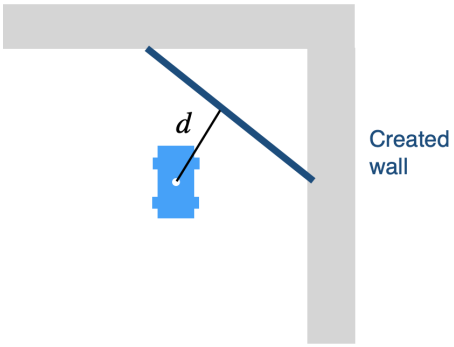

Figure 3: Artificial wall created by fitting a line to the front and side points combined. The car now measures its distance from the wall to be d, and this informs the error and steering angle that the car will take.

To fine-tune the turning on closed corners, we applied a multiplier to the steering angle produced by our PD controller. After we redefined the wall to cut across the corner, we calculated the error to this wall using the same method that we use for normal wall-following, and found the steering angle by passing in this error to our PD controller. However, the steering angle was not drastic enough to fully turn on tight, closed corners, so we introduced a steering_factor parameter that multiplies the steering angle by a constant. We found that steering_factor = 2.0 was strong enough to create successful turning on 90-degree closed corners and avoid colliding into the walls.

2.3 Safety Controller

Author(s): Rachel

The racecar contains numerous expensive sensors and components that can get damaged, especially when driving using the wall follower. To avoid collisions while the racecar drives autonomously, we implemented a safety controller that runs in parallel with the wall follower package that prevents the racecar from crashing into obstacles. Our ideal safety controller will enable the racecar to successfully achieve all the requirements of an efficient wall follower (drive close to walls, turn around corners, achieve high speeds, etc) while also be able to prevent crashes. To achieve this, we made a safety package that subscribes to sensors and intercept published driving commands, and publishes to the safety navigation topic if the racecar is in danger of collisions.

2.3.1 Hierarchy of Driving Controls

When the racecar drives autonomously using the wall follower package, publishing to the safety navigation topic will allow us to override the drive commands and prevent collisions due to the intrinsic hierarchy of driving controls presented by the muxes. We were able to achieve levels of control through publishing to different priority topics. For our racecar, there are three main essential navigation command topics each with different priority levels as described in Fig. 4. Since the safety topic has higher priority than the navigation topic, any driving commands published to safety will override any driving commands from navigation.

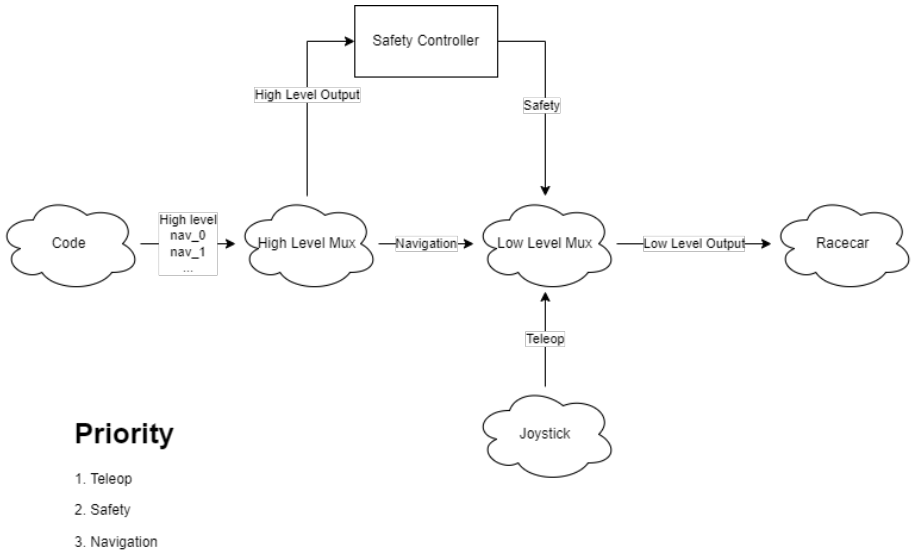

Figure 4: Graph of the procedure of drive commands and how code is transferred to the racecar. The priority of the three types of drive commands are ranked from teleop to safety to navigation where navigation is the drive command from the wall follower code.

2.3.2 Safety Controller Procedure

Our safety controller uses similar technical features as our wall follower. We subscribed to the LiDAR sensor data to get the distances the racecar is from objects in the environment. Since collisions occur when the racecar drives into an object, we sliced the LiDAR ranges data such that we only use the distances from the front of the racecar. This translates to -3 degrees to 3 degrees of the LiDAR data. After slicing the data, we underwent a series of steps to make the data usable for our safety controller.

1. Filter the data such that only distances within a range is accepted.
2. Did linear regression on the remaining data.
3. Calculated the distance the robot is to the line representing the wall.
4. If the distance is within a threshold, override the drive commands from the wall follower.

The range for the filter is determined by finding the minimum distance in the sliced data and adding an upper bound of two meters plus the minimum. This eliminates outliers that are very far away from the robot which prevents the calculated distance from being further than the actual distance. Using minimum

distance with an upper bound allows us to get a distance that is close to the racecar while not causing the racecar to be too cautious. This results in a safety controller that can detect sudden obstacles in front of the racecar in the form of objects or humans regardless of the speed the racecar is driving. All these steps put together enabled us to create a safety controller that successfully prevent collisions while allowing the racecar to maintain the ability to effectively follow the wall.

3 Experimental Evaluation

Author(s): Brian

3.1 Qualitative Evaluation

After implementing our wall follower and safety controller, we performed qualitative testing of our car's PD controller in order to quickly evaluate many different values for K_p and K_d and find the best set of gains for our car. We decided to use our gains from simulation as a starting point ($K_p = 5, K_d = 0$). This is what we defined as our untuned controller. We also wanted the car to follow the same wall for all the tests in order to reduce external factors in the car's performance. We selected a corner in Stata basement that was challenging enough to be insightful but also representative of a typical wall scenario our car might encounter as shown in Fig. 5. To optimize our gains, we started with values of K_p and K_d, recorded our observations about the car's smoothness, overshoot, and whether or not it successfully completed the turn, adjusted K_p and K_d accordingly, and then repeated the process until our car met our qualitative performance criteria for corners. These criteria were the following:

1. The car should successfully complete the turn without hitting the corner
2. The car should initiate the turn at a reasonable distance from the corner
3. The car should minimize overshoot
4. The car should minimize oscillations after completing the turn

A selection of some of our test points and observations are included in Fig. 6. Note that for these tests, the car followed the right wall at a distance of 1m.

Open Corner

Figure 5: Corner used for testing PD controller gains (yellow line indicates desired path.)

K_p	K_d	Observations
5	0	Lots of oscillations even when just following wall leading up to turn.
1	0	Went really far away from wall but did complete the corner eventually.
2	0	Completed turn in acceptable manner
2	0.35	Very steady when following wall leading up to turn, but completes a wide turn that eventually straightens out.
2	0.5	Tighter turn and oscillations are more dampened.

Figure 6: Select Kp and Kd values with observations.

From our testing, we observed that increasing K_p made the car respond quicker to changes in the wall but increasing it too much lead to undesired oscillatory

behavior. In general, increasing K_d seemed to dampen out the oscillations and smooth out the car's steering but too much K_d made the car sluggish. Thus, our approach to adjusting the gains was to first increase K_p until we noticed oscillations and then increase K_d to smooth out the oscillations.

Finally, we also evaluated our car on a more complex wall shape. To do this, we selected two walls in both Stata basement and Baker that combined various types of corners and wall geometries, including gaps in the wall, doorways, pillars, and more. The car successfully completed both courses so we deemed our controller robust enough to handle real world wall following scenarios.

3.2 Quantitative Evaluation

In addition to collecting qualitative data, we also recorded error measurements from our car as it was driving using both our original, untuned controlled ($K_p = 5, K_d = 0,$) and our new, tuned controller ($K_p = 2.25, K_d = 0.5,$) on three different scenarios: a closed corner, an open corner, and a simple, curved wall. These scenarios are depicted in Fig. 7. Recall from section 2.2.3 that we defined the error at any given time step as the difference between the desired distance from the wall and the actual distance from the wall. We then plotted these errors in order to compare our tuned and untuned controllers in Fig. 8, 9, and 10. Note that for these tests, the car followed the right wall at a distance of 0.5m.

Closed Corner | Open Corner | Simple Curved Wall

Figure 7: Test Scenarios (yellow line indicates desired path, orange cone denotes car's starting position).

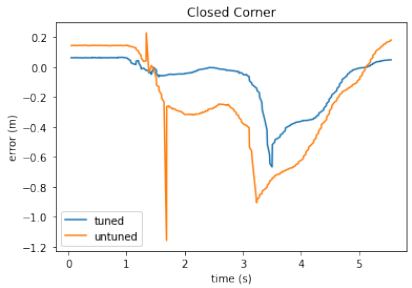

Figure 8: Error for closed corner scenario

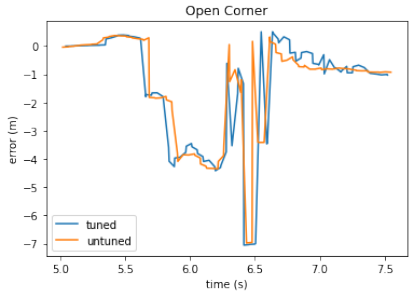

Figure 9: Error for open corner scenario

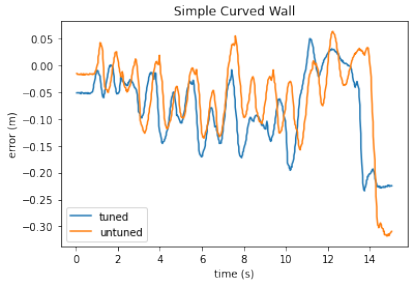

Figure 10: Error for curved wall scenario

In Fig. 8, we see that there is a big improvement for closed corners using the tuned controller. For example, around 3s which is when the turn occurs, the car had a maximum error of -0.7m for the tuned controller compared to -0.9m for the untuned controller. However, for the open corners and simple curved wall scenarios as shown in Fig. 9 and 10 respectively, while the car navigated

the scenarios successfully, we did not see as big of a difference in performance between the untuned and tuned controllers. This was unexpected given that our qualitative observations showed that the tuned controller was better, but there are a couple reasons we hypothesize our error plots aren't showing a definitive difference in performance. The main reason is that because we define our error relative to the visualized wall, which is itself derived from a noisy measurement, any error in the visualized wall means that our distance from the wall error will also be incorrect. Ideally, our error would be based on a ground truth wall and car position so that we would be able to calculate the true error. This idea is explored further in the conclusion section of this report.

It is also interesting to note that our error for the open corner is pretty high for both tuned and untuned controllers, reaching almost 7m at around 6.5s. During the test, we did observe our car overshoot the corner, but it never exceeded more than 0.5m from the desired following distance. While we were unable to diagnose a reason for this large measured error, it didn't affect our controller and in some ways was actually beneficial because it encouraged the controller to make a more extreme correction and thus turn the car faster. We plan to continue looking into this issue in future work.

We also calculated a sum of squares error error, see Eqn. 4, for each of the tests which is summarized in Fig. 11. While other error metrics also work, we wanted the squared term on the error because it penalizes large deviations from the desired wall following distance which is desirable for our wall follower. Again, the error for our tuned controller was significantly improved in the closed corner case but higher than our untuned controller in the open corner and simple curved wall cases. It is important to note that these numbers are only relevant in the context of the specific tests we ran and while they do suggest that we can make further optimizations to our K_p and K_d gains, we were not overly concerned as other aspects of the car's performance such as robustness and smoothness that were greatly improved using the tuned controller were not reflected in these error statistics.

$$\sum_{t=start}^{end} error(t)^2 \tag{4}$$

	tuned	untuned
Closed corner	4.8	12.2
Open corner	66.5	58.8
Simple curved wall	3.4	2.6

Figure 11: Sum of square error for tuned and untuned controller in test scenarios

Overall, our quantitative evaluation largely reinforced our conclusions from our qualitative tests. It also raised several areas for future investigation including figuring out a more accurate method for measuring error, rerunning the open corner tests to understand why the error was so large, continuing to tune our PD controller, and calculating different summary statistics to help quantify other aspects of our wall follower such as smoothness of the driving.

4 Conclusion

Author(s): Lauren (Summary, Next Steps), Brian (Limitations)

Overall, our car managed to successfully follow the wall at a desired distance including turning around corners and wall contours, and was able to stop and correct its steering when approaching people, objects, and other obstacles. This process was achieved through analyzing the LiDAR data by partitioning and filtering, tuning the steering angle using PD controller, and creating a safety controller. However, through our extensive experimentation and testing, we also learned about the limitations of our approach and areas that we can improve on in the future.

4.1 Limitations of distance to wall as a state variable

From the experimental evaluations, one of our conclusions was that using distance from the wall as the input to our controller has some drawbacks. While this metric is relatively simple to compute, the main problem is that there are infinitely many walls which share the same distance from the car as shown in Fig. 12. This is not ideal because if the car is approaching a closed corner like the one in Fig. 12, our wall detection logic would see the corner as the series of red lines as visualized in the figure. However, the car's distance, d, to each of the lines would be the same. This is why for closed corners, we had to write logic to force the car to start turning otherwise the PD controller would think the car is always at an acceptable distance from the wall and never 'see' the corner coming until it is too late.

A better approach might be to use the car's x, y position as the state variables for the PD controller. Instead of defining a desired distance from the wall, we would have the car pursue a unique x, y position that is at the desired distance from the wall. Picking this x, y position would require us to know where the car and walls are located within a global frame, which we would could accomplish using localization algorithms such as Simultaneous Localization and Mapping (SLAM).

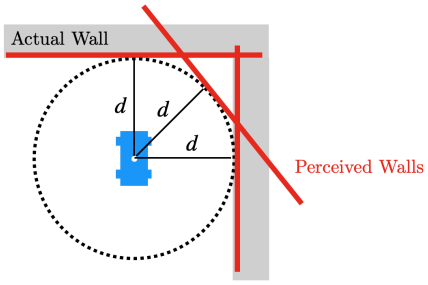

Figure 12: All walls tangent to circle with radius d centered on the car have the same perceived distance

4.2 Next Steps

In the next lab, our team will learn how to use our racecar's camera so that our car can park at certain markers. Specifically, we will do the following procedures.

1. Experiment/Prototype with several types of object detection algorithms.
2. Learn how to transform a pixel from an image to a real world plane using homography.
3. Develop a parking controller to park your robot in front of an orange cone.
4. Extend your parking controller into a line following controller.

These steps, along with our wall follower code, will enable the racecar to more efficiently drive autonomously. In particular a line follower working in conjugation with a wall follower will enable the racecar to handle different environments and minimize error.

5 Lessons Learned

Lauren:
I found that my team generally worked very well together. I felt that we had

good communication in terms of scheduling meetings, explaining concepts and algorithm implementations and in general treated each other with respect. From a technical standpoint, I had difficulty implementing my wall following algorithm in the previous lab, and working with team members who had more success with the algorithm, I learned a great deal about partitioning the LiDAR data and the motivations behind making certain divisions, in addition to determining the constant values for the PD controller. I believe that the greatest difficulty we had in completing this lab was **time** and **robot availability**. As multiple pod groups needed our robot for extended periods of time (and often overlapping time periods) to complete algorithm testing and troubleshooting before the lab deadline, coordination became difficult and frustrating and many of my team members had to work 20-30+ hours over the course of the week to complete the assignment by the due date.

An Bo:
The wall follower lab was quite an interesting and unique experience. For most of my project-based classes at MIT, we usually focus solely on the technical respective of the project objectives. This lab is my first time showcasing and communicating a team's technical results in the form of presentations and lab reports. My role during the lab was to implement and revise our wall follower code from the previous wall follower simulation lab into the real racecar. Interesting enough, we found when we tried and used each team member's initial wall follower code into the racecar, the racecar would perform completely differently than in the simulation. Following this discovery, I was tasked with rewriting our wall follower such that our wall following logic were implemented and various desired parameters could be easier modified without the need to dig into the source code. An important technical lesson I learned is to always modularize and document your software for future use and simulations may not always reflect reality. In regards to communication and collaboration, I feel lucky to be in a team from wide background since a lot of the team had some intuition on PID controllers and heuristics on optimizing K_p and $K - D$ parameters such that the racecar smoothly follows the wall. As well, as we tested the racecar, we were about to collectively contribute and communicate on whether we need to increase and decrease various parameters, modify our software logic, and plan for future meetings.

Brian:
From a technical perspective, one of the lessons I learned was that it is challenging to replicate results in simulation on the real racecar. When we were first starting this lab, we thought that we would be able to port over the wall following code from whoever had the highest score from the simulation lab. However, no matter whose code we tried, the car's behavior would be completely different than in the simulation often times steering and crashing in unexpected ways. In order to make our actual car successfully wall follow, we had to combine ideas from multiple teammates code. For example, the corner detection logic was drawn from Claire's code, the distance calculation formulas were from An

Bo's code, the usage of a front wall was inspired by Rachel and Lauren's labs, and the PD controller implementation was from my code. This combination of our ideas turned out to be one of the biggest benefits of working on this lab together. Working with other people also helped make the debugging process faster whenever we encountered issues. For example, when we just beginning to work on this lab, there were many instances when I would forget to run a command, like source devel/setup.bash, and get an error. With the help of my teammates collective wisdom, I was always able to quickly figure out where I made my mistake(s). The lab also helped me realize the importance of having a well organized team workspace. Our team set up a Notion workspace which proved to be invaluable in helping us keep track of deadlines, take notes and record data, and assign tasks to each member for the report and presentation. I was grateful to have teammates who helped keep the workspace updated and were also proactive in keep their own responsibilities organized. Overall, I had a lot of fun with this lab and look forward to the next lab!

Claire:
I do not come from a robotics background, so I definitely learned a lot technically in terms of translating simulation to real life, and how important it was to have ample time to test and tune the robot. It also forced us to think critically about the robustness of our wall-following algorithm, as there were perhaps certain methods that might have been good enough for the simulation, but failed in real life once we had much noisier and more inconsistent LiDAR data, as well as a much wider variety in the kinds of walls that we had to follow. I learned a lot about ways to figure out what was going wrong, and we heavily relied on RViz to identify why our wall-follower wasn't working early on in the lab. Thanks to An Bo, we were able to parametrize our code using rosparam and that was massively beneficial to being able to test different parameters quickly. I feel lucky that our team communicated very effectively with each other - we were always responsive in our team group chat, Brian set up an amazing Notion page that was super helpful for keeping us organized and storing knowledge / data as we worked on the lab, and we worked well with each other. I would say the biggest challenge was collaborating and communicating with the other teams in our pods, especially since we had to share cars. While we were on the same page as each other, it was sometimes hard to get ahold of other teams which resulted in us losing valuable time to work with the cars. Overall, the model of sharing cars definitely limits progress versus us having our own cars, but we are gradually learning how best to coordinate as a pod.

Rachel:
This wall follower lab was a very rewarding and challenging problem for me to tackle. I do have some background experience working with and programming robots through my UROP. However, my UROP project consisted of working with a robotic arm and not a movable racecar. This lab allowed me to apply all the programming and robotics skills I have from my prior experience, along with the skills I learned from the prior labs, to an intensive problem. I really

appreciated the buildup of labs and how all the prior labs really prepared me to tackle this wall follower lab. This resulted in this experience to be even more enriching and satisfying to complete. This lab really helped solidify the skills I learned working with ROS publishers and subscribers, as well as working with LiDAR data. Moreover, my communication and collaboration skills were definitely put to the test as my team and I tried to combine our simulation wall follower code and try to successfully run the code on the physical racecar. Not only did I learn how to communicate and divide tasks efficiently, but I further practiced my debugging and organization skills. I learned the value of having a well organized program for easier implementation and testing and how to utilize everyone's skill sets effectively. I believe my team worked well together. It was due to everyone's effort and will that we managed to successfully program a wall follower and safety controller. This paper and the presentation serves as the product of our teamwork, hard work, and perseverance.