# Lab #3 Report: Wall Follower

Team # 18

Cindy Wang
Eric Gonzalez
Herbert Turner
Jose Lavariega
Reinaldo Figueroa

6.141 RSS

March 5, 2022

## 1 Introduction (Herbert)

Transferring simulation code to work on actual hardware is an important step in the process of developing a physical autonomous system. The goal of this lab was to take the lessons we learned developing a virtual wall follower and create a wall follower for the physical racecar provided to us. The wall follower is given a side(left or right) and a desired distance, and uses calculations based on LiDAR data to adjust its heading and stay the desired distance away from the wall. It must be robust: able to navigate corners and successfully track the wall from far away. A safety controller also had to be implemented to ensure our race car avoids crashing even when the regular controller fails. Our implemented wall follower takes points from the relevant angle range of the LiDAR scan and uses the Least Means Square method to calculate the line best representing the included points on the wall. We use a PID controller to control our heading to approach the desired distance from the wall. Our implemented safety controller limits our maximum speed as a function of our distance to the wall and performs an emergency brake if we are too close.

## 2 Technical Approach (Cindy)

At a high level, the goal of this lab was to develop a system for our race-car to follow alongside a wall with a given distance from the wall and simultaneously check for obstacle collisions in the front. We divided our implementation into the following sections: parsing LIDAR scan data **(2.1)**, wall detection **(2.2)**, wall following control **(2.3)**, and safety control **(2.4)**. In addition, we will mention some of the most significant challenges and bugs we encountered during the initial tuning in section **(2.5)** while implementing and testing our wall follower controller.

### 2.1 Parsing LIDAR Scan Data (Cindy)

We receive LIDAR scan data in the LaserScan message format. Because not all of the data in the message is relevant to each following task, it is helpful to split the data into meaningful sections that we can then further manipulate. We divided the scan data into three sections: *front*, *left*, and *right*. The *front* section is relevant for the safety control because we are determining if there are any obstacles in front of the racecar. The *left* and *right* sections are relevant for detecting and following the wall to the left and right of the racecar, respectively.

After this sectioning of scan data, we only use the relevant part for the following computations.

### 2.2 Wall Detection (Cindy)

Given the scan data to the left or right of the car, we want to be able to detect or estimate where the wall is relative to the car. We accomplish this by using a least squares regression.

Because the format of the scan data gives us angles and their corresponding ranges, it is first necessary to convert the data to a Cartesian coordinate space, with the car's LIDAR centered at the origin. This was done simply with the following equations.

$$ranges = [d_1, d_2, ..., d_n] \; angles = [\theta_1, \theta_2, ..., \theta_n] \tag{2.11}$$

$$x_n = d_n * sin(\theta_n) \; y_n = d_n * cos(\theta_n) \tag{2.12}$$

We removed any outliers in the data by using a heuristic value maximum that eliminates a point if its distance is greater than some value set by parameters. If we had additional time, it would've been cleaner to use a RANSAC approach to remove outliers.

Then we used numpy's built in least squares regression function from `numpy.linalg.lstsq`. We chose this linear regression function as it is a widely used model, and for most of this lab we were testing with straight, linear walls. In the future, if we were to want to detect and follow a less homogeneous wall structure, or walls that incorporated more curves and corners, it would be more effective to use a more sensitive modeling method.

After using a model, we are able to determine a linear equation as an estimate for the wall with respect to the LIDAR. The vehicle can than evaluate how far it is from the wall it calculated in the linear equation.

## 2.3 Wall Following Control (Cindy)

We implemented a Proportional-Derivative (PD) controller for our car to follow the wall and maintain a given distance from the wall. A PD controller gives us flexibility to respond better to wall curves and contours.

For the error calculation in our PD calculations, we used the desired distance from the wall subtracted by the actual distance from the wall. The resulting output from our PD calculation would be used as the steering angle for the AckermannDriveStamped sent to the /drive topic.
The gains for the PD controller were tuned as we needed to modify them going from the simulation to the real car as described in section 3.3

$$d = \frac{\overrightarrow{p_2p_1} \times \overrightarrow{p_1}}{\|\overrightarrow{p_2p_1}\|} \tag{2.31}$$

$$e(t) = d_{desired} - d_{measured} \tag{2.32}$$

$$\theta(t) = K_p e(t) + K_d \frac{d}{dt} e(t) \tag{2.33}$$

$$K_p = 2, K_d = 1/20 \tag{2.34}$$

## 2.4 Safety Control (Cindy)

The final component of the lab was to implement a safety controller that prevents the car from crashing into obstacles in front of it. This was simply done using a heuristic check where we took the minimum range from the front section as calculated in 2.1. We then checked whether this distance was less than a certain value, and we would slow break or emergency break as necessary.

## 2.5 Initial Tuning and Challenges - Rey

Besides calibrating our PD controller which is also briefly mentioned in section **(3.3.1)** later on , we had to adapt our wall following controller from Lab 2 which was working with a range array that only contained 100 distance values. Therefore, we were unsure if our implementation would perform as good in our racecar as it did in simulation. In order to solve this conflict, we made our code able to accept LiDAR scan data with range arrays of variable size. This new method, in addition to the outlier rejection method mentioned in **(2.2)** also changed our way to calculate angles. At first, we used the same vision range for both sides, from 75 to 6 degrees, however, it did not perform as

expected. We realized that such range was not good enough to plot a line side by side with our racecar and use it as our reference from the wall, so we decided to pick new values that would better serve this purpose. After exhaustively testing different ranges, we finally arrived to the conclusion that a range from 120 to 15 degrees in both sides would be enough to satisfy that goal.
Throughout this process of tuning and testing, we unconsciously inserted a bug into our code that would basically make our line to be plotted wrong and therefore, the actual distance from the wall would be miscalculated. This bug came from the fact that we were performing outlier rejection after doing angle calculation, causing our ranges to not match their corresponding angle, and therefore describing a line unrelated to our data. Finding this bug was a whole team effort that required us to sit down and go over the code in detail. After it was fixed, our problem with the distance calculation was fixed, and our racecar performed as initially expected.

## 3 Experimental Evaluation - Jose

In this section, we show the systematic approach to choosing the best Wall Follower controller, as well as evidence the correct functionality of the wall follower through fast convergence to the desired offset from the wall. We show a multi-step approach using both simulation and real-world data. We start off by outlining the heuristics taken to determine the initial controller on top of which we built up the rest of our code, then continue on to evaluate additional controller architectures that ultimately got discarded in favor of the controller we ended up choosing. Subsequently the approach for tuning the controller through experiments on the hardware is done, as well as further simplifying our mathematical formulation and finally any legacy tools that we implement for further labs.

### 3.1 Initial Controller Selection - Jose

Our team had a choice of multiple controller implementations coming in from the first lab. Some of them were highly-tuned PD Controllers, while others attempted different tuner architectures like pure pursuit, but with moderate degrees of success. To choose the base building block for our controller, we compared performances across the submission for Lab 2. The highest scored controller out of the five would be what the code would be based on. Naturally, additional elements that worked particularly efficiently on other controllers could be added to the base controller. The table below shows our initial choice to go with Reinaldo's Controller as a base.

| **Controller** | **Score** |
|---|---|
| Jose | 96.3 |
| Reinaldo | 98.2 |
| Cindy | 89.1 |
| Herbert | 96.2 |
| Eric | 96.2 |

Table 1: Initial Controller Selection

Reinaldo's controller consisted of a PD with additional cost for turning tight turns in the 90 degree turn case. As observed in section 2, over time a more aggressive 90 degree turn override controller was implemented, as well as an aggressive safety controller for the velocity.

### 3.2 Evaluation of Additional Controller Architectures - Jose

A quick idea on the team was to evaluate possible different controller architectures, notably the pure pursuit controller. At the start of the lab, we lacked the capability to compare completely different controller architectures other than by running test cases from the Lab 2 test case and comparing equal performances. Test cases 4 and 5 proved particularly useful to qualitatively observe controller performance. Additional scripts developed in between Labs 3 and 4 will prove better for evaluating the controller architecture in simulator, although given complications as discussed in sections 4 and 5, these implementations could not be part of the scope of the team focus.

The pure pursuit controller was compared with test case 4, with a convergence $L_1$ (the look ahead distance) chosen from the candidates of 2.4, 1.7, 0.5 . The best results were seen for a candidate of 2.4, although at this point other prevalent issues started to be observed with the car and so the focus was kept to just the PD Controller.

## 3.3 Tuning the Controller and Experiments - Jose

Having chosen the PD controller, the team proceeded to tune the controller, which involved collecting data and systematically determining the rate of convergence. All of these tests were done on hardware, after choosing the general architecture from the Simulator candidate architectures. The first evaluation we did was a parameter space search for our PD controller. These tests were necessary because the inherent mechanical limitations of the race car meant that the controller had to be much more smooth than was required in simulation.

### 3.3.1 Initial PD Controller Tuning - Jose

While the initial gains for the implementation of Reinaldo's PD Controller were suitable in the simulator, they were not suitable in the real-world hardware. Much of the iteration on PD Gains was on firstly fixing the Line Visualization and interpolation bug as outlined in the Challenges section, as well as getting back minimal functionality to avoid crashing into a wall. Both were incredibly time consuming which left little room for iteration with multiple controller gains. In the end, a script was written that could directly analyze a trial run based on LIDAR distance estimates compared to the desired distance. The initial run with such a script is shown below.

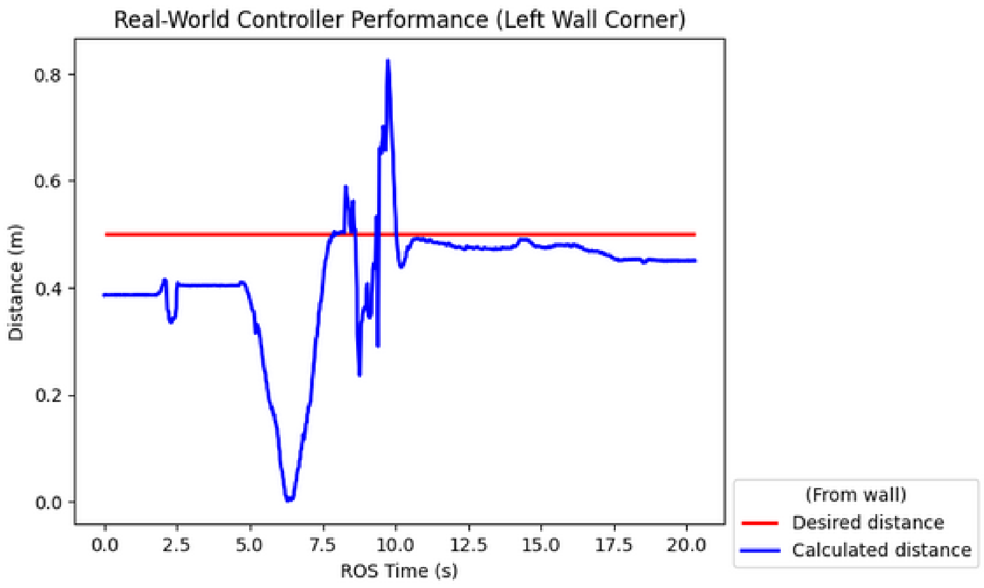

Figure 1: Initial Controller Performance with left Wall

The regions of abrupt local maxima and minima represent obstacles or open areas where the LIDAR measurement abruptly changes as it sees a wall different from what it is following.

### 3.3.2 Safety Controller Testing

For testing the safety controller, we approached a specific corner as seen in the figure below, and this was our standard for checking the safety controller. This wall segment poses a challenge as it requires to turn left into an area that requires a tighter turn to get out, while it also sees a pillar that can throw off wall measurements from the front. A good safety controller should be able to get out of such a situation while not crashing.

### 3.3.3 PD Controller Tuning - Jose

For testing, the team uses two metrics to evaluate the performance on a controller, the Loss and the Score. They are both defined below:
**Loss:**

$$\sum_{i=1}^{n=1} \text{Desired distance} - \text{Actual Distance}_{t_i}$$

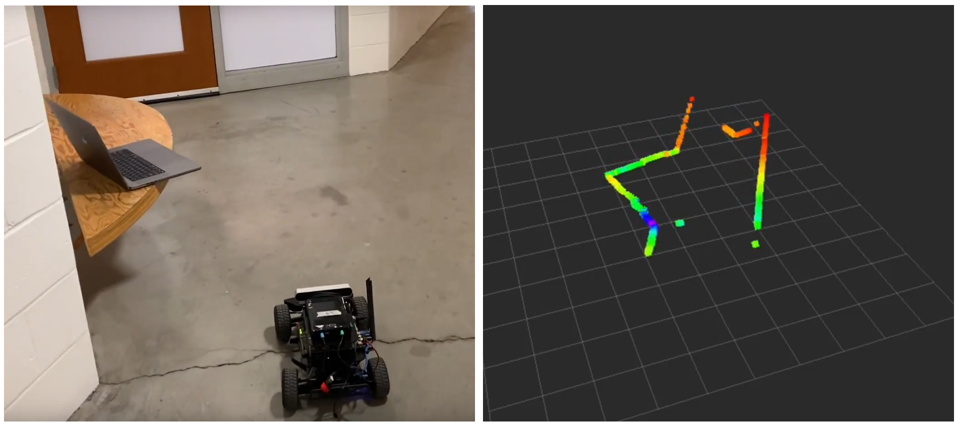

Figure 2: Edge case region for wall follower safety controller. The tight left and right turns as well as the pillar makes this difficult to navigate.

**Score:**

$$\frac{1}{1 + (4 * \text{loss})^2}$$

Using the Loss and Score metrics, across multiple trials the PD Controller, the results from those bags are shown as the remaining figures in this section:
Following a wall and encountering an open hallway, to a desired of 0.75 m.

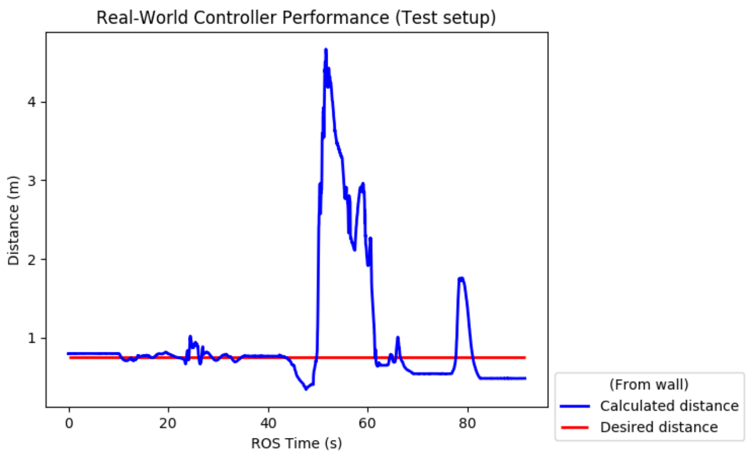

Figure 3: cornerOpenHallway bag
Controller: PD, Safety: norm, Loss:0.382468992251 Score: 0.299354548438

Subsequently, the team recorded Rosbags on further runs, and started to build up a library of ideal scenarios that we would want our car to be subject to, to accurately claim that an appropriate controller has been chosen. Below, there are multiple trials, each represented by a ROSBAG, that show the final controller performance on a multitude of scenarios.
Across each of these ROSBAGS, regions with a lot of noise were prevalent.

## 3.4 Legacy Tools for Further Labs - Jose

The team's lab had to be shifted in scope by the steep learning curve associated with porting to hardware as well as multiple hardware issues that prevented a fully systematic approach to testing controllers and informing decisions on gains. However, what the team does takeaway is a start in development for a systematic way to test multiple controllers by logging results from the test cases in Lab 2, which allows quick iteration on multiple architectures.

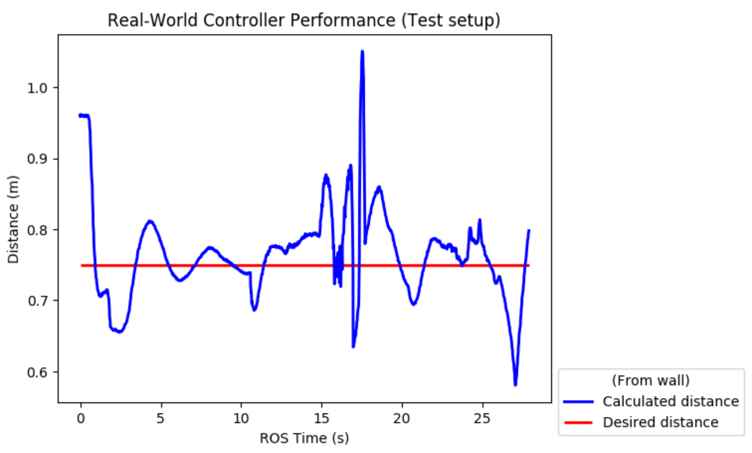

Figure 4: RightTurn Kp1 Kd0.1 bag
Controller: PD, Safety: norm, Loss:0.0468701613

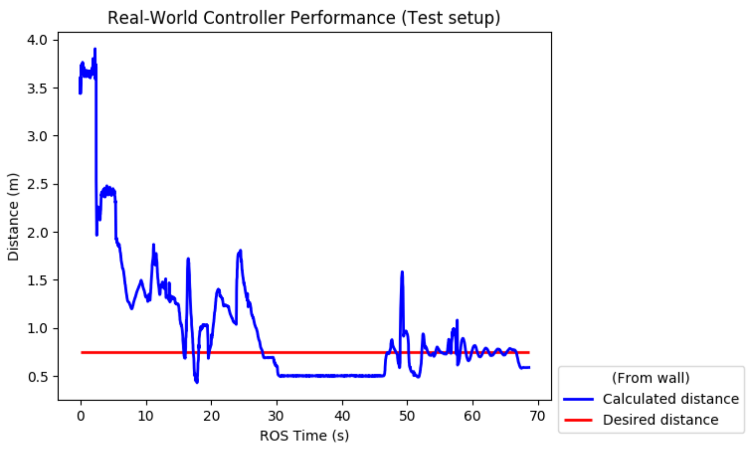

Figure 5: Kp2 Kd 0.1 bag
Controller: PD, Safety: norm, Loss:0.440804621382 Score: 0.243371803745

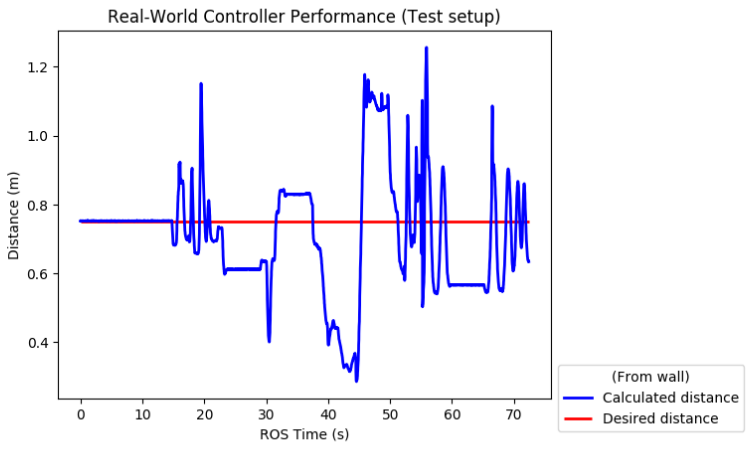

Figure 6: Noisy Lidar Measurement bagController: PD, Safety: norm, Loss:0.127106335547 Score: 0.794599060237

In addition to scripts running with the simulator, our team also created multiple ROSBAGs and a systemic way to record and save their data, which will be useful for further analysis. The team did manage to run ROSBAGs and analyze with one of the final controllers, as well as the safety controller.

## 4 Conclusion (Eric)

Overall, this design phase taught the team how to transition from simulation to running on actual hardware. The goal of the lab was to implement the virtual wall follower onto actual hardware and a safety controller, which we managed to achieve to an acceptable degree. In the ideal scenario, the virtual wall follower implementation would have worked with minimal changes made to the code. However, the team ran into a variety of problems as a result of transitioning to physical hardware. Much of the learning with respect to this lab was a result of these problems. In a sense, the team learned how to operate the vehicle and the problems that can arise from transitioning to physical components.

After learning how to operate the vehicle with a joystick controller, the team worked on fine tuning the parameters for the PD controller, as well as resolve bugs with respect to detecting physical walls. Initially, the team would visually evaluate the vehicle's performance running along walls. As experiments continued and became more elaborate, visual evaluation proved to be an insufficient measurement tool. As a result, we transitioned to collecting information from ROSBAGs.

Following the use of ROSBAGs, we were better able to quantify performance, and so the team was able to continue development of the wall following system. Once at a satisfactory development phase, the team also constructed a safety controller to autonomously prevent the vehicle from crashing. This was possible due to experience detecting walls. Overall, the team was able to revamp the virtual wall following implementation and make it more robust to real world hardware and scenarios. For future design phases, being able to detect walls may serve as a starting point for detecting other objects. Finally, a safety controller would allow the vehicle to safely test future labs without fear of damage.

## 5 Lessons Learned

### 5.1 Cindy

In this lab, I worked on setting up the car and it's connection for the first time alongside Eric. There were some technical problems involving the car not being able to automatically connect to the router without the use of a monitor as well as with the joystick controller not connecting. We solved this by getting to lab early and asking a TA for help. I learned that when I am blocked by something, it is worth spending some time to investigate, but after some time it is more efficient to move on to something else and ask a TA for help. I also learned about how important it is to record rosbags so that we can more systematically tune and modify our controller. In the beginning, we relied on our eyes to determine what changes needed to be made, but this is not scalable for when there are more degrees of freedom and the problems become larger to solve. I also learned that in practice, sometimes heuristics are a quick hack to solve a problem, and while it may not be the most technically clean method, it does often get the job done. For this first team based lab, I learned how getting stuck in the implementation details or issues can not be very time effective. Rather, it is more valuable to have a foundation to work off of first. Finally, I learned the importance of communication, stepping up to help my teammates whenever there were issues, and overall showing up for each other. From answering questions in the group-chat, to helping store and carry the car, to showing up to all group meetings, to working offline with teammates to debug, these were all some softer skills that I got to work on to help build the team culture!

### 5.2 Eric

This lab, I began by working on configuring the vehicle alongside my labmate Cindy. We learned how to power the on-board computer and the motor. From there, we ran into issues connecting the computer to a router. This router would allow us to connect to the on-board computer and make modifications as needed. Therefore, it was critical that we resolve how to connect the vehicle to a router. Furthermore, the joystick controller, while not necessary, also had connectivity issues. We felt uncomfortable moving on without a joystick to help us save the vehicle from accidents. Thus, we sought out help from Teaching Assistant's. We learned to ask for help rather than sit around and waste time.
Another technical skill we began to develop was how to use version control systems, such as Git. While not perfect yet, the team now has exposure to using Git, as well as the benefits and problems that can arise. I believe we will only improve in the future, so that we can avoid running into problems when collaborating on files together.
Lastly, working with the team has been a learning experience too. For such a complex assignment and material, it

was crucial we learn to divide up the work, as well as come together to collaborate. It is important we communicate to ensure our tasks are completed with few hiccups. After all, the tasks are complex enough. We are fortunate to have practiced on teamwork this lab.

### 5.3 Herbie

The first thing I worked on was creating the safety controller package. It consists of a single node that calculates the distance of the car from the closest obstacle on its current trajectory and outputs the corrected, ”safe” velocity for the car. I also worked on cleaning and modularizing the wall follower code. Through this lab I learned how important clean and modular code is in the team setting. The setup of my safety controller allowed Jose to work on various safety policies in parallel and easily swap them in and out of the controller. The modularization and readability improvements of the wall follower code made it easier for us to find bugs and get it working properly. The last technical piece I worked on was our analysis code. This was a script that read in the desired distance from the wall, actual distance from the wall, and time data and generated the loss,score and a plot of desired distance and actual distance over time of a bag file. Working on this part of the lab, taught me the importance of standardizing evaluation metrics to help make informed design decision as well as the importance of well-presented data.
On the communication side, I also learned a lot about working in a team through this lab. I learned how to communicate my ideas properly to other team members. I also learned about organizing tasks to allow us to work in parallel. This allowed members not to be blocked in their work by other members, so everyone was able to contribute equally to the project. Lastly, I got useful experience in giving group presentations. I learned the importance of planning an, as well as how to create a good transition. I also got a lot better at

### 5.4 Rey

In this lab, I worked mainly on improving our wall following algorithm by modularizing and using more efficient methods for multiple calculations . I also worked on improving the way we used the scan data from the LiDAR to get our line plotted in Rviz. In addition, I spent a considerable amount of hours working on calibration of our PD gain values with multiple members of the team until we got a good performance. I learned a lot about how to use Github for version control since I never had the chance of working with a team using the same Github repository in this type of projects. Cindy helped me a lot to become comfortable using git for branching which allowed me to leave behind my bad habit of making major changes directly on the racecar. For once I started to get familiar with the ease of knowing how to test new code on different branches and switching from one branch to the other in the racecar. I also learned that whenever I have a bug on a portion of code that I wrote myself and I am unable to find where the bug is, it is better to let your teammates know what is happening, and go through the code with them. Sometimes I find the bug by reading the code out loud, or my teammate catches it as I explain what the code does. Also, communication in our team has been extremely good and has avoided a lot of confusion that I have had in previous teams. Everyone knew what they were supposed to do and who to go to if more details on how a different portion of the controller works. This has been just the first lab and I feel like I have become a better team player and that I can be more useful to the team, beyond individual skills. Regardless of the difficulties we encountered, I am really happy with the work we have put into our racecar, and I am sure the best is still yet to come.

### 5.5 Jose

From the start I was working on the safety controller, from an algorithmic perspective. I started off trying different approaches in the simulator and when I arrived to some behavior that I was satisfied with, I started porting it over to Herbie’s ROS node. Furthermore several more of the algorithms in my code ended up being passed over, as I had used a parametrization in my LIDAR preprocessing for Lab 2, as well as some aggressive control specifically for making turns whenever there was a wall in front of the car. I served more of an algorithmic role as well when tunin gains and thinking about which conventions we had taken in the code initially that were no longer valid for the real-life hardware platform. I think where I learned the most was in implementation issues that are not apparent in the sim but are in the real car, and certainly I am taking more care in the next lab to have a satisfactory behavior with my parking controller quickly to be able to port over to the real car faster. I ended up working on multiple bug fixes and issues that came along as we were tuning the controller. I shared my code for analyzing individual ROSBAGs with Herbie, so it did end up that a lot of the algorithms I wrote and existing code I had was used. Certainly next time I will take more care of making my code much cleaner to read, and focus on the real-life implementation on the car much faster. Another big one is to be rigorous from the start, and log the decision process behind any heuristic or decision we take in a more systematic manner. We will be recording ROSBAGs from the beginning for next lab.