Lab #3 Report: Wall-Following on the Racecar

Team #1

Alex Ellison
Kyri Chen
Ashley Holton
Jordan Ren
Laura Rosado (editor)

6.141: RSS

March 5, 2022

1 Introduction (Kyri)

In this lab, we have designed an autonomous wall-following algorithm for the racecar and adapted it for use in real-life scenarios. Furthermore, we implemented a safety controller to prevent crashes into obstacles and minimize damage to the racecar and its associated equipment. Since we had previously created a wall-following algorithm for a simulation environment in Lab 2, our initial work in this lab focuses on adapting the existing code to overcome differences between the simulated and physical racecar, such as the difference in the lidar data collection and reporting, and a noisier real-world environment.

Our first challenge was adapting the scan-processing code to a new lidar scanner model. In contrast with the fixed scanner on the simulated car that provides a constant array of distances between angles $[-\frac{2\pi}{3}, \frac{2\pi}{3}]$, the Velodyne lidar scanner provides more comprehensive coverage at the cost of incomplete data at each scan. In order to use the complete Velodyne lidar scan, we need to combine and preprocess groups of two scans. Our complete methodology will be discussed further in section 2.1 of the report.

Next, we found that there were a variety of confounding variables in real life which required us to reiterate on and improve our wall-following algorithm. As we tested our original code in real-life settings like dormitories and hallways, we found that rotated lidar reference frames made it harder to calibrate the robot, open doors made it more difficult to accurately detect the wall, and narrow corridors required adjusting the clipping of scan ranges. In order to create a

more robust controller for maintaining a set distance to the wall, we utilized a standard PD controller with an extra gain relating to the slope of detected walls in order to round sharp corners better, which we will elaborate on in section 2.2.

Lastly, to protect the racecar from damage during testing, we designed and implemented a safety controller that constantly runs in the background of the robot's navigation code. Whenever the car comes within a specified distance of an obstacle in its path, the safety controller overrides any autonomous control, in this lab the wall-following algorithm, in order to bring the car to a stop.

2 Technical Approach

2.1 Wall-Follower (Alex)

A critical piece of any autonomous system is goal-oriented control. In the case of our lab this week, our goal was to follow a specified wall at a desired distance while traveling at a certain velocity. In order to achieve this goal we developed a slightly modified PID controller that relied on lidar data as external input.
Our system consisted of three main components: Data Cleaning and Segmentation, Wall Detection, and Augmented PID Control. The following subsections will go into more detail on our implementations of each of these pieces and give some insight into our decisions relating to each. A video of the racecar successfully following the wall can be found here.

2.1.1 Data Cleaning and Segmentation

In simulation we were given access to clean, full lidar data. Unfortunately, in real life, the data isn't as clean. Instead of receiving a message containing a full scan from the minimum to maximum angle of the lidar scanner, we instead get incomplete and offset data. As seen in Figure 1, the physical lidar scanner mounted on the car returns distances at angles offset by 60 degrees in the negative direction (depicted by θ) from the reference frame of the racecar, and does not cover the full scan range in each output message. We were able to empirically discover that in order to utilize a full scan of data, it was necessary to synthesize the data between two consecutive scans together as shown in Figure 1.

Mechanically, our process of cleaning the data consists of the following:

1. Define a flag ($flag = False$) which will indicate when we have received LaserScan data from the lidar scanner

2. Subscribe to LaserScan data output by the lidar scanner on the scan topic. Once we receive data, set $flag$ to True and store the full scan we received in a variable ($previous\_scan$)

3. Wait until we receive data again. Now that $flag = True$, instead of storing the scan data, loop through both the current scan ($current\_scan$)

Figure 1: lidar Offset and Incomplete Data

data we just received as well as *previous_scan* and define *full_scan* as the pairwise minimum of *previous_scan* and *current_scan* divided by two since the scanner measures the distance to the wall and back and we only care about the distance to the wall. Then erase the data stored in *previous_scan* and go back to step 1

This method functions to combine pairs of consecutive scans together and remove most of the gaps in scan values which manifest as infinite floats.

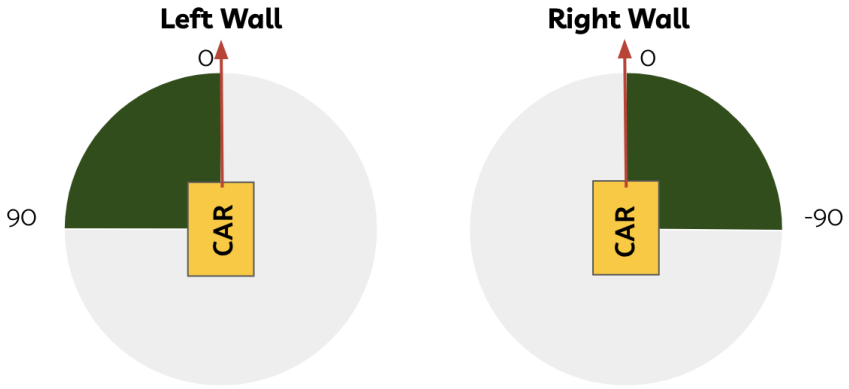


Figure 2: lidar Segmentation

Finally, while following walls on one side of the robot, it makes sense to segment the area of scan data we are looking at. Since it is only necessary for the robot to "see" the wall on its specified side to follow it successfully, we opted to look from 0-degrees to 90-degrees while following the left wall (measured from the reference frame of the car) and from -90-degrees to 0-degrees while following

the right wall. These ranges are shown in Figure 2. We also determined that, in our current iteration, data received from very far away from the robot is not useful because we want to follow the wall closely, typically around 1 meter away. To account for this, we only looked at scan data within 3 meters of the vehicle. Using this logic, we filtered down our input data even more and accounted for the lidar offset described earlier using the following method:

1. Define two arrays to store our segmented scan distances and angles (*scan_segment* and *angle_segment*)
2. Define start and end angles for whichever side we want to follow (*start* and *end*) and offset by the lidar offset described earlier to make sure we grab the correct data for the reference from of the car. Ex. For the right side $start = 90 + lidar\_offset$ and $end = 0 + lidar\_offset$.
3. Loop through all the scan data which we cleaned earlier ($full\_scan$) and their corresponding angles ($full\_angles$) and append all data points that lie within the range defined by *start* and *end* and are less than 3 meters away.

At the end of our cleaning and segmentation methods we are left with two nicely formatted arrays containing ranges data and the corresponding angles. The next section will discuss how these data are utilized to implement our controller and achieve our goal of autonomously following a wall.

2.1.2 Augmented PID Control

In this lab we opted to use a simple PID controller with a slight modification to include an extra gain. In order to do this, we used linear regression to define a line from our lidar point scans and then use that line to calculate an error term as shown in Figure 3. We then experimentally optimized each of our parameters.

As shown in Figure 3, the desired distance to the wall is subtracted from the perpendicular distance between the racecar and the wall to calculate our error term. This is interpreted as how far away the car is from the desired following distance to the wall. In practice, the math consists of finding the distance between a point (that of our car's lidar scanner (0, 0)) and a line (our wall). To calculate this we define our wall line using two points (x_1, y_1) and (x_2, y_2) taken from converting the lidar scan data from polar to Cartesian coordinates, running them through our linear regression model, and plugging them into the following equation:

$$SIDE * \left(\frac{\|(x_2 - x_1) * (y_1 - 0) - (x_1 - 0) * (y_2 - y_1)\|}{((x_2 - x_1)^2 + (y_2 - y_1)^2)^{\frac{1}{2}}} - DESIRED\_DISTANCE\right)$$

Where $SIDE$ is defined as 1 if we are following the left wall and -1 if we are following the right wall and $DESIRED\_DISTANCE$ is the how far away from

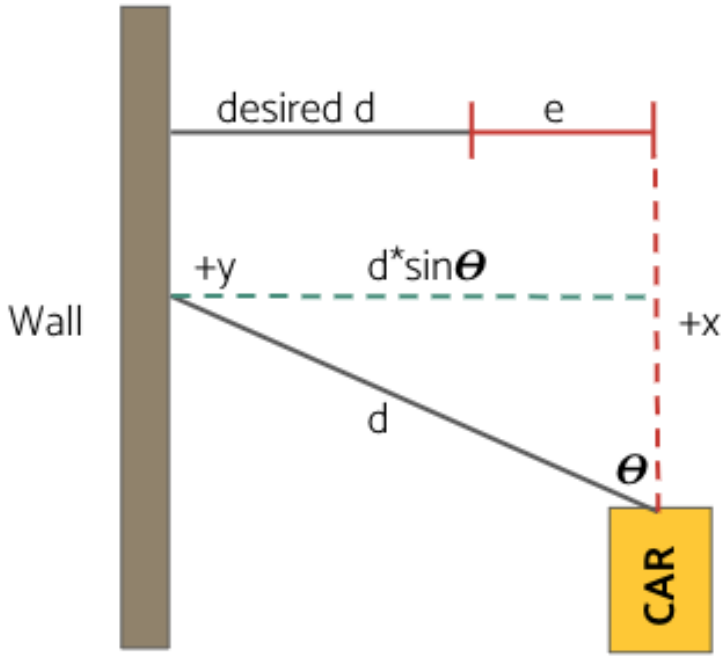


Figure 3: Error Calculation Based on Angled lidar Scan

the wall we want to be. After tuning our PID controller in simulation we observed decent performance with little to no oscillation using proportional and derivative control, but the car struggled to turn sharp corners. To amend this, we added a gain that is multiplied by the slope of the calculated line that attempts to make the car parallel to the wall. This noticeably improved our car's performance when rounding corners.

Figure 4 illustrates the function of this gain (K_s) while following the left wall. When the car is turned away from the wall, the slope of the perceived wall is positive with respect to the reference frame of the car, and thus the slope term will induce a positive turn, or left turn, back towards the wall. When the slope is negative, or that car is turned towards the wall, the slope term induces a negative, or right turn, away from the wall. While the car is parallel to the wall, the slope is 0 and K_s will have no influence on the controller.

After defining our PIDS controller we were left with the following equation to define our control:

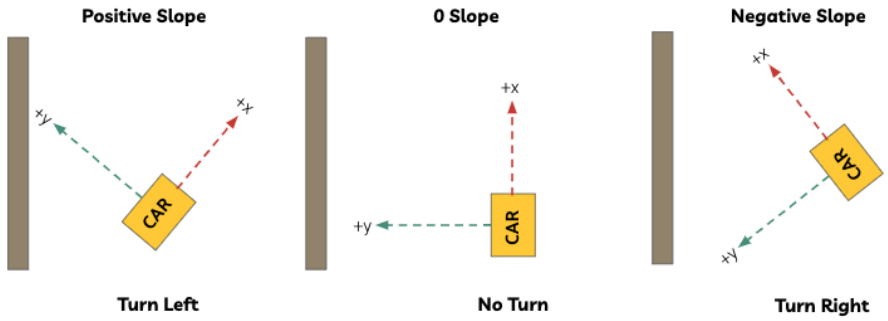


Figure 4: Slope Augmentation for PID Controller

$$steering\_angle = K_p * e + K_i * i + K_d * d + K_s * s$$

Where e is the error, i is an integral term defined as the integral of the error over time, d is a derivative term defined as the change in error or time, and s is the slope term defined as the slope of our perceived wall line.

By observing the wall-following performance of the racecar robot in simulation and in real life, we arrived at a PDS controller with $K_p = 1.5$, $K_d = .05$, and $K_s = .5$. Overall, this performed well and we were able to follow the wall adequately. An evaluation of the final performance is discussed further in Section 3.1.

2.2 Safety Controller (Kyri)

Unexpected behavior during testing and implementation is an inevitability in any project. We implemented a safety controller that is run simultaneously to other behaviors to predict impending crashes and stop the racecar to avoid damage to the hardware.

Our approach prioritizes simplicity and robustness. At a high level, the safety controller constantly scans a "crash zone" in front of the racecar, stopping if it detects any obstacles in the zone. The size and shape of the "crash zone" is determined by the steering angle and a parameter specifying how cautious or risky the safety controller should be. A video of the safety controller successfully overriding the wall-following algorithm can be found here.

As shown in Figure 5, the "crash zone" is rectangular when the car is driving straight forward and trapezoidal when the car is turning. While the most accurate "crash zone" would be the projected future path of the car, we thought

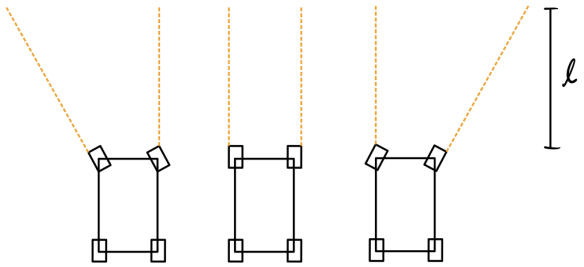


Figure 5: Crash zone for each case

that implementing Ackermann steering would be overly-complicated for a basic safety controller. The region formed by the orange lines in the figure form an outer-bound on the projected path of the car, allowing for simpler calculations without sacrificing too much accuracy.

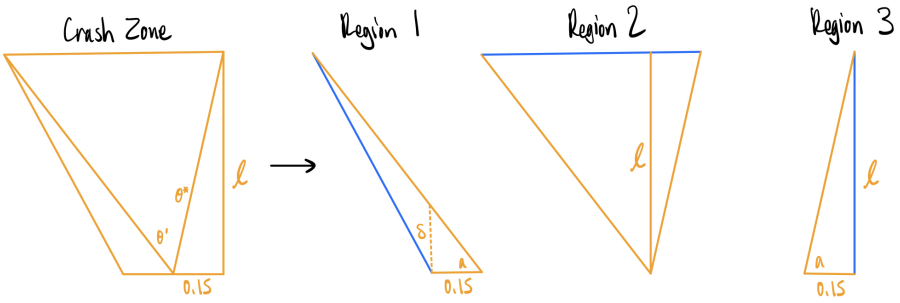


Figure 6: Left-turn "crash zone" calculations

Since the lidar scanner is not situated at the outer edge of the wheels where distances are easily calculated, the boundaries of the "crash zone" need to be measured in polar coordinates with the scanner as the origin. To examine the specifics of the calculations, we investigate the left-turn "crash zone," which is decomposed in Figure 6. The symmetric nature of the other two "crash zones" allows us to apply the same mathematical concepts to their cases.

In Region 1, which is between scan angles $\theta \in (\theta', \frac{pi}{2}]$, the distance from the scanner to the boundary highlighted in blue, denote it r, can be calculated using the Law of Sines. Since we know the steering angle δ, we can calculate the angle across from side r, denote it b, is

$$b = \frac{\pi}{4} + \delta.$$

Furthermore, angle a is complementary to the scan angle, so we know that

$$a = \frac{\pi}{4} - \theta$$
$$c = \frac{\pi}{2} - a - b = \theta - \delta.$$

Applying the Law of Sines with the measured scanner-to-wheel distance of the racecar, we find that the distance from the scanner to the "crash zone" boundary in Region 1 is given by

$$r = \frac{\sin b}{\sin c} \cdot 0.15.$$

Next, in Region 2, or scans with scan angle $\theta \in (\theta^*, \theta']$, the distance from the scanner to the boundary highlighted in blue can be calculated from the observation that

$$r \cos\theta = \ell$$
$$\implies r = \frac{\ell}{\cos\theta}.$$

Lastly, in Region 3, or scans with scan angle $\theta \in (\theta', -\frac{\pi}{2}]$, the distance from the scanner to the boundary highlighted in blue can be calculated from the observation that

$$a = \frac{\pi}{2} + \theta \qquad \text{since } \theta < 0$$
$$r \cos a = 0.15$$
$$\implies r = \frac{0.15}{\cos a}.$$

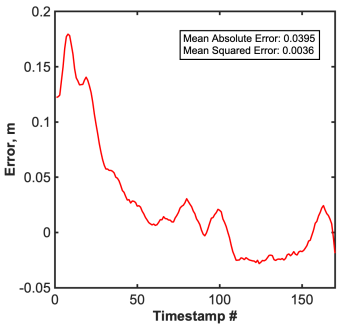
To use this scan data as a safety controller, the lidar scanner publishes a signal to the car to stop whenever it detects an object within the boundaries of the "crash zone." We chose this design for our safety controller because it is simple enough for the car to calculate quickly before advancing too far into the "crash zone" and risking a crash.

3 Experimental Evaluation

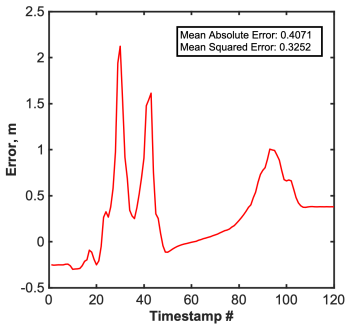
3.1 Wall Follower (Laura)

We evaluated the performance of the wall-follower code by collecting and graphing the error over time for two scenarios: a straight course and a two-turn course. Visualizing the change in error over time and under different circumstances helps us understand our controller's performance and gives us qualitative metrics to make improvements in the future.

In Figure 7a, we see that the error quickly decreases from the initial conditions. Each timestamp corresponds to a fraction of a second. Since we were limited by the length of the wall in the area we were testing, we were not able to collect data to identify a period of oscillation. Had we observed a constant oscillation, we



(a) Error measurements over a straight course.



(b) Error measurements over a two-turn course.

Figure 7: Error graphs for the wall-following experiments, collected while following the left wall at a desired distance of 1 meter.

would consider increasing the derivative gain to increase damping. In addition, since the mean absolute error only represents a 4% variance from the desired distance of 1 meter, there is no need to introduce an integral gain to improve steady-state error. In this scenario, our controller performed adequately.

When a turn is added to the course, the reported error will inevitably increase. This is because the calculated line is unable to precisely track a corner, and since the actual distance is calculated from this line it leads to a spike in the error graph. Figure 7b highlights this phenomenon. There are two spikes corresponding to each of the two terms. Since the corners were in rapid succession, the racecar did not have time to settle to the low error values we observed over the straight course described earlier. While the mean squared error ($0.3252m^2$) is two order of magnitudes higher than what was reported for the straight course ($0.0035m^2$), this is not a concern because the peaks are essentially outliers. By our qualitative assessment, our controller was sufficient to navigate this course.

3.2 Safety Controller (Ashley)

To test the effectiveness of our safety controller, we would drive the car with either the wall-following code, or some simple code to drive forward at a set velocity, then introduce an obstacle to the car's path. We tested with a variety of obstacles, including a plastic bin, a person's legs, and a cardboard brick. We made sure to test with obstacles of different sizes, shapes, and materials to ensure that the safety controller was robust enough to handle different types of objects it could encounter in the real world.

For all kinds of obstacles, the car was able to stop within the specified stopping distance. We observed on some runs of the experiments that the car would jerk

a bit back and forward as it came to a stop. We theorize this may be due to the inertia of the car tilting the lidar towards the ground as the car comes to a stop. We addressed this issue by slowing down the robot when it's getting close to the minimum distance specified to be away from obstacles.

4 Conclusion (Jordan)

In this lab we have successfully designed and implemented an autonomous wall-following algorithm for our racecar that is both robust and efficient. We first find the wall on the correct side of the car by parsing lidar scan data and putting it through a linear regression model. Then, the car is sent steering instructions based on a modified PD controller. Using this method the racecar was able to find a wall even if not placed directly next to one, and follow it through various types of turns.

Furthermore, we implemented a safety controller that stops the car when an unexpected obstacle appears in front of the car. The safety controller uses lidar scan data in regions in front of the car in order to decide whether or not to stop. To make the stopping more smooth, we also implemented a slow-down mechanism so the car does not come to a stuttering stop. When any obstacle suddenly appears in front of the car, or if the wall-following algorithm gets the car stuck the safety controller will step in to keep the car from crashing.

While the current iteration of our wall following algorithm works well, we could potentially reduce our mean squared error by making our turns more efficient. This could be done by converting our model into a pure pursuit controller, which draws a line a constant distance away from the lidar data and attempts to stay exactly on the line. Another improvement that we could make is tuning the PD and K_s gain values to reduce the steady-state error of our wall follower. Finally, we also need to set and tune a gain value for our minimum collision distance for the safety controller. This would increase the distance at which the safety controller starts to slow down or stop as the speed of the car increases.

5 Lessons Learned

5.1 Kyri

Technically, I learned that a simpler approach will often get the job done, and I should spend time understanding the problem statement and avoid over-complicating it. At first, I implemented code to avoid obstacles instead of a simple safety-controller, resulting in a lot of time spent on a solution that wasn't necessary. I learned that communication and collaboration is best done when scheduling as a unit. Since some pieces (ie. figuring out the Velodyne lidar scanner) were necessary in order to accomplish parts of the lab, we could've communicated/prioritized more effectively to get those parts done.

5.2 Ashley

Through this lab, I learned more about methods to deal with real-world data and noise. I also got to practice how to communicate about technical projects in a clear and concise manner through the presentation. I learned more about how to use animations in an efficient way, and that sometimes, less is more. When it came to collaboration, I know that I need to be more proactive in future labs in figuring out what part of the project I should work on and coordinating when we can all work together on the project.

5.3 Alex

This week was full of many learning opportunities for me both technically and with respect to communication. First, I learned that planning is just as important as execution. Looking bad at this weeks technical challenges, much of the difficulties I faced could have been alleviated by devoting more time to planning out a course of attack and writing out pseudo-code and math before starting to program. Second, I found that pair-programming was an effective way to collectively contribute ideas to our system while also minimizing errors. This is something I will definitely be taking more advantage of in future weeks. Finally, I picked up meany tips and tricks about making presentations from my teammates who have had more experience with technical presentations than I. Some of the highlights were how to add slide animations to highlight what I would be talking about, utilizing more graphics than words, and looking up slide templates to minimize the amount of time spent searching for icons and graphics. Overall, this week presented me with many challenges that allowed me to learn technically and become a better communicator.

5.4 Jordan

From a technical standpoint I found that planning out a well-structured layout for any piece of code will help make the debugging process more simple. For instance, when working with the lidar scans it was extremely hard to understand in code how I should shift our reference frame. However, with the help of some drawings and collaboration from team mates the problem was eventually solved. By consulting others before implementing any technical solutions I could have saved myself a significant portion of time in this lab. In terms of communication and collaboration I learned that having a fresh set of eyes on any problem can help progress the lab. There were many times where I was stuck on an issue due to assumptions I have made, where simply having someone question everything you do can highlight any flaws. For the next lab I will certainly ask for help and work with others more often.

5.5 Laura

I learned many things over the course of this lab, both on the technical and communication side. Processing the lidar data and tuning the PID controller

stick out to me as some of the larger hurdles we had to overcome in this lab. Since I have more of a background in dynamics and controls, in hindsight I could have been more active in helping to solve these problems earlier. Knowing when to take advantage of our relative strengths will be key to our success in this class. This was also my first time working on code collaboratively, and I realized both the importance of writing a good commit line for git as well as being able to explain my rationale to my teammates. Now that we're familiar with the hardware, I hope we can be more intentional in delegating tasks and planning our workflow. I think it will not only help us solve technical issues faster, but also organize the information we want to present in briefings and reports.