Lab #3 Report:
Wall-Follower and Safety Controller

Team #1

Jensen Curtiss
Nick Dow
Dane Gleason
Sadhana Lolla
Sienna Williams

RSS Spring 2023

March 11, 2023

1 Introduction

Sienna Williams and Nick Dow

The final challenge of RSS is to program an autonomous racecar to race around a track in the basement of Stata and compete against our peers. This final challenge is daunting if one were to jump straight into it, so the course is structured to slowly build up our understanding and the capability of our robot through a series of labs with escalating requirements.

In order to be able to race our car around a closed loop track, it will need to be able to follow straight lines and park in front of objects. To build up to these capabilities, Lab 3 requires us to build wall-following capability into the racecar. In Lab 2, we wrote code to allow a car to follow a wall at a constant speed and distance away from it in a simulation; it was required to be robust enough to turn corners and follow uneven walls in a simulation. This wall-follower code included a Proportional-Integral-Derivative (PID) controller as well as a front-wall-detector system. Implementing this basic behavior gave us an introduction to control theory and the practical considerations needed to make a robot autonomous.

Building on this foundation, our goal in Lab 3 was to put this code on the actual racecar and test its capabilities. The front wall detector was added so

that the racecar would be able to maintain the desired distance when turning an inside corner. The wall-follower code also needed the capability to turn outside corners as well as right itself from various orientations that are not aligned with the nearest wall. After adjusting PID control gains, the PID controller from Lab 2 was able to quickly realign the racecar to follow the wall without inducing oscillations that a normal proportional controller would experience. However, we observed that as the velocity of the car increased these parameters were less effective and more oscillations were observed. We tested the wall-following algorithm by running it on multiple types of walls at different distances and speeds while measuring the error observed. Throughout the development and testing process, we expected and did run into multiple hardware hurdles, necessitating modifications to the simulation wall-follower code to adapt it to actual racecar.

The racecar is extremely expensive with some of its components costing thousands of dollars. In order to protect the equipment, part of Lab 3's technical requirements is to construct a safety controller that will prevent the racecar from crashing into objects that could damage itself or those objects. While this safety controller should be robust, it should not be so conservative that it limits the normal functionality of the racecar. Despite additional hardware constraints in the form of incomplete lidar data, we implemented a safety controller that strikes the balance between these conflicting objectives. The safety controller was tested with two main tests. Firstly, we placed the racecar such that the wall-follower algorithm cannot avoid a wall collision, therefore causing the safety controller to ensure that the car stops before hitting the wall. Secondly, it was tested by having a team member repeatedly step in front of the racecar and observing it stop when obstructed and start again afterward.

Overall, we present a successful wall-following and safety controller system that can handle a variety of environments, including but not limited to acute and obtuse corners, divots in walls, and high speeds.

2 Technical Approach

Sadhana Lolla, Nick Dow, Sienna Williams, Dane Gleason

In this section, we present the methodologies and technical details necessary to develop a wall-following algorithm and safety controller. The goal of the wall-following algorithm is to control the car steering angle θ to minimize the constraint $|d_{wall} - d_{desired}|$, holding velocity, $d_{desired}$, and all other car parameters constant. We define d_{wall} as the distance measured by the lidar sensor between the car and the wall. We achieve this by building a PID controller that minimizes the distance of the car from an algorithmically constructed wall that reduces noise and outliers from a real wall.

Secondly, we develop a safety controller that ensures that $||p_{car} - p_{object}||_2 > d_{desired}$ for all obstacles, where p_{car} is the position of the car and the p_{object} is the position of every object in the car's environment.

As mentioned previously, the focus of this lab is to implement the initial autonomous capabilities of our car in building toward a fully autonomous racecar. Specifically, since we are transferring autonomous wall-following simulation to real race car robots, developing and testing the algorithms above presents several challenges because real hardware is much more error-prone than the simulation's abstractions.

2.1 Velodyne lidar Sensor

Sadhana Lolla, Nick Dow, Dane Gleason

The cars that we used for Lab 3 contain a Velodyne lidar scanner, which is quite different from the lidar scanner used in simulation. Therefore, before we were able to move our wall-following simulation to the robot, we first had to preprocess the Velodyne scans to a format that our algorithms could digest. Unlike the Hokuyo scans and the scanner in simulation, the Velodyne lidar scanner outputs 3D data, which was converted to 2D data by the staff. However, each packet only contains half of the original data, and the rest were populated by infinite values. We first diagnosed this issue by deploying our code onto a Hokuyo lidar car, and we found that the Hokuyo lidar immediately integrated with our previous code to produce correct wall-following behavior. While the Velodyne's incompatibility was certainly a setback, our opportunity to test on another car was important as it allowed us to discover that the issues in our implementation stemmed strictly from the lidar configuration of the car with all other factors being equal. To convert the Velodyne data into a representation we could use, we combined both packets of data. Although the configuration of the Velodyne data resulted in halving the frame rate of our input data, it was a necessary step and had no impact on wall-following accuracy.

Even after fixing these issues, we noticed that some of the lidar data was always infinite. We first attributed this to the fact that the Velodyne has 360 degrees of sensing, as opposed to the Hokuyo and simulation lidar sensors which had only 270 degrees. Therefore, at some points the Velodyne senses the robot itself, which results in infinite values. Our initial solution was to populate these infinite values with the furthest distance seen so far so that they were never included in the wall-following algorithms. Later, we also realized that when the robot gets close enough to a wall, within 0.5m, the measurement here also becomes infinity. The implication of this discovery means that our initial solution causes the wall to effectively disappear to the robot, causing the robot to not be able to turn corners. We solved this by filtering out the range of angles that correspond to the Velodyne sensor looking into the robot body and then populating any infinite values with the minimum distance seen in that scan, as

we knew that these values corresponded to the wall being within the minimum detectable distance by the Velodyne. We also had to adjust parameters in our wall-following and safety controllers to account for this constraint.

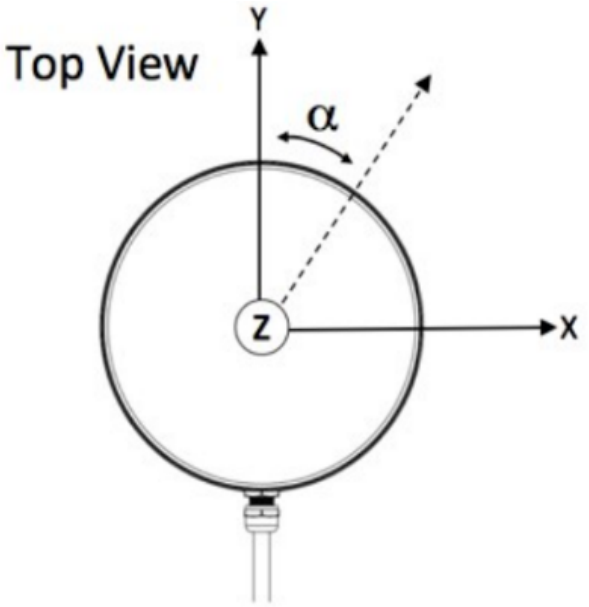

Figure 1: Top Down view of the orientation of the Velodyne lidar sensor. True front of the Racecar had an α of $-60\,\text{deg}$ with respect to the y-axis of this figure due to the sensor's mounting position.

In addition, the lidar scanner was rotated by 60 degrees to the right on the racecar body. Figure 1 illustrates the Velodyne lidar coordinate frame and conventions which proved to be very useful in debugging and understanding the data our Velodyne was publishing. We realized that the visualization software we used, Rviz, visualizes the frame of the robot and the laser scanner as the same orientation, while in reality the laser is offset by sixty degrees, as shown in Figure 1. To resolve this mounting misalignment, we rotated the total data scan array to line up the frame of the lidar scanner with the robot frame.

2.2 Wall-Following Algorithm

With these initial hardware challenges addressed, we were able to deploy the implementation of our wall-follower. We utilized the base algorithm from our highest scoring team member's code.

2.2.1 Angle Filtering

Sadhana Lolla

At every timestep, the lidar scanner provides the robot with a set of distance measurements associated to a range of angles. As mentioned above, we pre-process these data packets to remove the half of the data that only sees the robot, and then we aggregate packets together to form a complete lidar scan.

Our first goal is to detect a wall that is parallel to the robot itself. We do this by considering only the portion of the lidar scan that faces either the right or the left wall, as shown in Figure 3. For example, using the right wall configuration the robot would only use the lidar scans corresponding to angles ranging from $\frac{\pi}{6}$ to $\frac{\pi}{2}$. We also remove outliers from these points, since the line detected will be incorrect if distances that are too far are used to construct it. We define outliers as points within the desired angle region where $d_\theta > 2 * \min(d_{desired}, d_{min})$. The reasoning behind this filter is that if a measurement in the desired angle range is larger than twice the minimum distance or twice the desired distance, it is likely too far from the robot to be part of the wall we actually want to follow. Therefore, we do not include it in further data processing.

Once we can detect a wall next to the robot, the next step was to enable the robot to turn when it is faced with a corner while continuing to maintain $d_{desired}$ from the wall. We experimented with many different variants of corner and front-wall detection, but found that a large range of angles ended up creating a very noisy estimate of the wall. Therefore, we use the angles ranging from zero to $\mathbb{1} \times \frac{\pi}{6}$ to determine if there is a wall in front of the robot, where $\mathbb{1}$ is an indicator variable that is -1 if we are following the right wall and 1 if we are following the left wall. We choose this narrow region since it enables more robust corner detection: through experimental trials of running the wall-follower, we found that using a larger range of angles that is adjacent to those that we use for wall detection results in often-incorrect corner detection. We theorize that the points in the region zero to $\frac{\pi}{6}$ often belong either to the front wall or to the side wall, and it is ambiguous as to if there is actually a corner present. Therefore, we only use the points in this region if we have detected a wall in the front wall detector region shown in Figure 2.

2.2.2 Linear Regression

Sadhana Lolla

Once we have filtered outliers and determined which points constitute the line we want to follow, we find the line that best fits this list of data points. We cannot use these data points directly: walls typically contain curvature, divots, or other anomalous features that we want to smooth out so that the car is able to maintain a smoother trajectory. To do this, we use linear regression with the following formula:

$$\theta = (X^T X)^{-1} X^T b \tag{1}$$

Equation 1. *Linear Regression*

θ is a 1×2 parameter array, where the first value represents the slope of the predicted line and the second value is the intercept, or bias. b is the original y-coordinates of the data points we use to construct the line. To calculate the regressed Y, we can multiple $X \times \theta$. When the robot is faced with a corner, the linear regression becomes useful as well: the points that make up the corner regress to a diagonal line. By maintaining a distance of $d_{desired}$ from this diagonal line (shown in Figure 2), the robot can avoid hitting corners or turning too late, which would occur without the linear regression method.

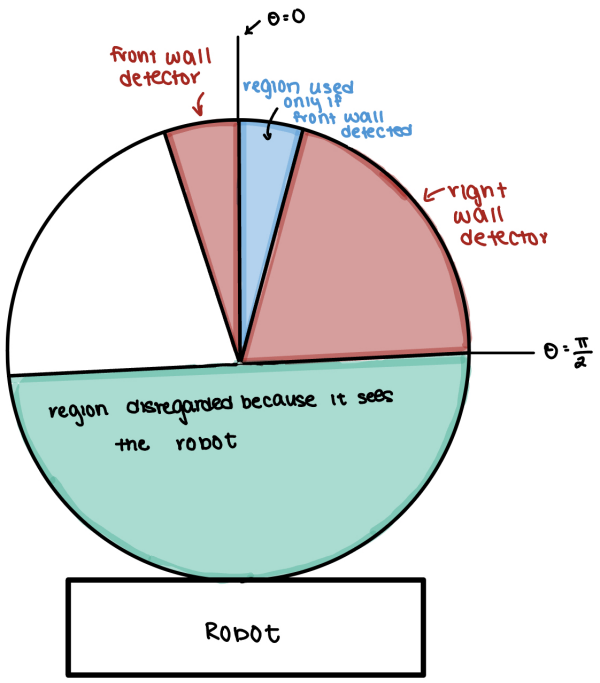

Figure 2: Regions of lidar scan that are used to determine corners or walls when we are detecting the right wall. The same is true for detecting a left corner (reflected over the $\theta = 0$ line).

2.2.3 PID Control

Sienna Williams, Sadhana Lolla

The wall-follower uses a PID controller to keep the racecar a consistent desired distance away from the wall on one side. The PID controller determines

the steering angle at every timestamp, and is critical for the robot's ability to handle corners. The following equation shows the generalized form of PID:

$$u(t) = K_p * e(t) + K_i * \int_E e(t)\,dt + K_d * \frac{de(t)}{dt} \tag{2}$$

Equation 2. PID Controller

The error term $e(t)$ is $|d_{wall} - d_{desired}|$, where d_{wall} is the minimum distance between the robot and the regressed line calculated earlier. This distinction is crucial– by using the distance from the regressed line instead of the actual data points, we build resiliency and robustness into our wall-follower. We use proportional control to minimize this error, integral control to prevent long term error, and derivative control to prevent the car from oscillating.

The gains on the PID controller needed to be changed significantly when moving this algorithm from simulation to hardware. We turned the K_p, K_i, and K_d parameters were tuned based on experimental measurements. The K_p value determines how fast the racecar could get to the desired distance. The K_i value determines how much of the steady state error was eliminated. The K_d value determines how sharp the car would turn and allowed us to dampen the oscillations. The values that worked best were $K_p = 1.5$, $K_i = 0.75$, $K_p = 0.75$.

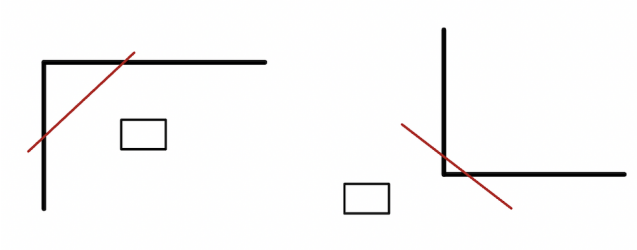

Figure 3: (Left) An example of the linear regression line in red when the car is going towards an acute corner. (Right) An example of the same when the car is going towards an obtuse corner.

2.3 Safety Controller

Nick Dow, Sadhana Lolla

Throughout our wall-following development, the car would occasionally crash when tuning PID parameters or when experimentally determining the limitations of the lidar. To combat the potential damage inflicted on the robot, we developed a safety controller that minimized damage to the car by overriding the wall-following behavior when a collision was deemed imminent. Our

decision-making throughout the development of the safety controller was informed by the dual objectives of preventing damage to racecar and not inhibiting the autonomous performance of the car in normal conditions. By being too conservative, the race car would not be able to track any wall very closely, but not being safe enough may damage components.

As discussed earlier in this report, the Velodyne outputs an infinite distance measurement for any measurement below approximately 0.5m: this approximation was discovered through observational testing. This hardware unreliability poses a problem for the controller as the controller is completely oblivious to whether a 'infinite' measure means a wall is 0.5m or 0.1m meters away. Our idea to combat this issue is to set every infinite measurement to the minimum distance value in the lidar scan. This pre-processing allows us to process the data normally and gives us a conservative estimate on how close the car is to any obstacle.

2.3.1 Controller Design

After confirming that the wall-follower prevents the car from crashing into wall on its sides, we decided that the safety controller would analyze the data directly in front of the car and prevent head-on collisions. The safety controller design we decided on is simple but effective. Within an arc of distance in front of the car, we check if there is an obstacle that is returning a distance measurement less than the `DESIRED_DISTANCE` parameter specified in a parameter file. As the lowest measurement we can actually receive from the Velodyne is 0.5m and this distance is already quite far to be stopping, we were forced to use 0.5m as our desired distance. We experimented with settings for the arc length through observational testing, and landed on π for the range of the sensing arc centered in front of the racecar. As directed in the lab instructions, these safety commands were published to higher priority topics than the navigational commands so that the safety controller would override the wall-follower navigation when necessary.

2.3.2 Integration Limitations

The limitation on the quality of the data we received from the lidar prevents our safety controller and wall-follower from working together at low wall-following distances. We found that we could only run the safety controller with wall-follower desired distances that were greater than a meter.

3 Experimental Evaluation

Dane Gleason, Sienna Williams, Sadhana Lolla, and Jensen Curtiss

3.1 Procedure

Dane Gleason, Sadhana Lolla, Jensen Curtiss
In order to ensure that the racecar can effectively maneuver a variety of scenarios that it might encounter on the racetrack, we tested the wall-follower implementation on both sides of the robots as well as various distances, speeds, and types of walls.

The safety controller was tested upon successful implementation of the wall-follower with a goal of preventing crashes. As mentioned earlier, this was inherently limited by our hardware, with our testing procedure informed by this limitation as well. Specifically, the distances we were able to test were set to minimize the effects of having a relatively large minimum sensing distance. As with the wall-follower, various scenarios were tested including trajectories towrard walls, static objects, and dynamic objects.

To ensure our testing was robust in characterizing performance, we ran the wall-follower code with varying parameters and environments while simultaneously recording a rosbag file that we could analyze later. With the obtained rosbag data, we were able to augment our qualitative testing with quantitative data. The data recorded included the desired distance from the wall and the actual distance from the wall at every time step to be fed into an error calculation.

3.2 Data

Sienna Williams, Dane Gleason
From the data that was collected through the rosbag, we calculated the average error and standard deviation for each test. The equation for the calculation of the error at each time step is show in Equation 3.

$$Error = \frac{|d_{wall} - d_{desired}|}{d_{desired}} \tag{3}$$

Equation 3. Error Calculation

We would've liked to have more time to process data properly, notably for cropping the data. When starting rosbag recordings, a constant error is observed corresponding to the delay between wall following initiation and recording. This results in an overall increase in error and affects both the average and standard deviation metrics, though this was relatively consistent for all recordings such that we are still confident in our overall observed trends.

3.3 Performance Metrics

Dane Gleason, Sadhana Lolla
Our metrics of performance for the wall-follower are the average error and the standard deviation of the error as defined by equation 3. We believe that these

metrics present an intuitive way of understanding the performance of our wall-follower from a control perspective. From this control perspective, the desired distance parameter is our set point and the error is the difference between the input and response of the system, so evaluating the error will be a direct indication of the performance of our control system.

Because the error varies over time, taking the average error allowed us to determine on average how far away the car strayed from the the desired trajectory. While helpful as a point performance metric, the average error can be skewed and will not shed light into how well we follow corners or other scenarios. Therefore, we also evaluate the standard deviation of the error, which allows us to quantify the consistency of our wall-following algorithm. A high standard deviation indicates an inconsistent performance while a low standard deviation indicates more consistent performance. We note that both of these performance metrics are selected considering the wall-follower exclusively and may be subject to interaction with the safety controller implementation with consequent misrepresentation of performance.

Our metric of performance for the safety controller is much more qualitative as the relative frequency of unnecessary stops. While we did not count these occurrences explicitly, this metric is more of a binary as to whether the controller is under- or over-protective.

With our metrics established, we set a goal of less than or equal to 20% average error for wall-following performance. Since equation 3 takes the absolute value of the difference of distances, this means 20% error to either side of the desired trajectory is defined as acceptable. We note that a percentage metric is subject to our desired distance, as a larger desired distance results in smaller error percentage for the same deviation from the trajectory. Also note that this effect is contrasted (though certainly not balanced or mitigated) by our car's increased oscillation at larger desired distances.

3.4 Results Analysis

Dane Gleason, Sadhana Lolla
Our experimental evaluation showed that our wall-follower and safety controller were functionally successful, though each could use improvement. We found that at smaller distances, we could not run the safety controller due to the Velodyne lidar constraint but the error was much lower than when we tried to follow the wall at a larger desired distance.

As shown in figure 4 and 3.4, as the speed increases, we saw an increase in error. This was expected because when the racecar is moving faster it has less time to correct itself causing higher errors. If gains were adjusted, we would expect to see a smaller error in these scenarios. Despite this, we still saw rela-

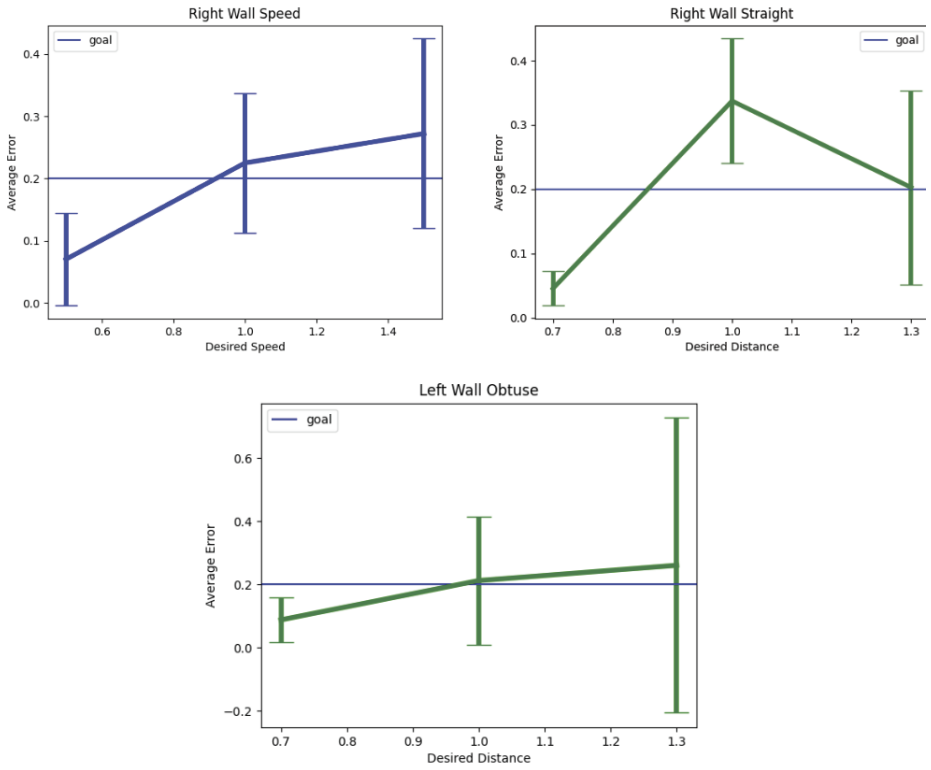

Figure 4: (Top Left) Graph of average error when testing different speeds on the same stretch of wall. (Top Right) Graph of average error as desired distance from the wall increases on a straight wall. (Bottom) Graph of average error as desired distance from the wall increases when turning an obtuse corner.

Distance (M)	Side	**Mean**	**Stdev**	Side	**Mean**	**Stdev**
0.7		0.153	*0.120*		0.049	0.027
1.0	Left	0.295	0.205	Right	0.223	0.157
1.3		0.421	0.220		0.339	0.093

Table 1: Average error when following a straight wall

Distance (M)	Side	**Mean**	**Stdev**	Side	**Mean**	**Stdev**
0.7		0.272	*0.452*		0.263	0.271
1.0	Left	0.093	0.072	Right	0.022	0.023
1.3		0.222	0.205		0.079	0.076

Table 2: Average error when following an obtuse corner

tively acceptable average error with respect to our goal in these tests with the average error being 0.295 at a speed of 1.5 m/s.

As shown in Table 1, our wall-follower followed straight walls with very low error at a threshold of 0.7. As the desired distance increased, the error also increased, and we attribute this to increased noise in the environment being picked up by the lidar scans, especially since we tested in relatively dense locations such as the Stata lab room. However, when accounting for the standard deviation, all of these tests fell within the 20% goal threshold that we specified earlier.

When following an obtuse corner, we actually found that a distance of one meter from the wall works the best. We attribute this to the Velodyne not being able to sense distances less than half a meter. Through qualitative measurement, we noticed that the robot tended to turn very at corners when $d_{desired}$ was 0.7M since it always thought the wall was 0.5M away, when in reality it was much closer. This has a larger impact on turns, which we see here.

We found that the wall-follower did not handle acute corners very well, for the same reason that it could not follow obtuse corners at small distances. Due to the Velodyne sensor, the robot tended to overshoot the corner and change steering angle too late. We found that this occurred regardless of $d_{desired}$, and the results are documented for $d_{desired} = 1.0$ in Table 3.

Side	**Mean**	**Stdev**
Left	0.310	*0.202*
Right	0.448	0.088

Table 3: Average error when following an acute corner at $d_{desired} = 1.0$

Speed	Mean	Stdev
0.5	0.075	*0.078*
1.0	0.219	0.111
1.5	0.295	0.205

Table 4: Average error when varying speeds and following the right wall at $d_{desired} = 0.7$

We would like to note that both through data analysis and observation that our wall-following implementation is overly sensitive to initial conditions. Oscillation with varying severity is observed in testing and depending on the initial pose of the car, sometimes results in failed corners or unnecessary stops with the safety controller activated. This is certainly something we hope to improve with future iterations and suspect the cause is under-damping of our system.

We observed that the car did not crash into walls while the safety controller was employed. This is an important result as it means that we can operate autonomously with confidence, though further testing may be required for very high speed operation. We were able to show that the safety controller can handle fast-moving dynamic changes in environment: when a human steps in front of the robot, it pauses immediately, and when the human moves out of the trajectory, the wall-follower code continues immediately. A key finding is that the safety controller unnecessarily stopped the car frequently. With this, we consider our controller to be over-protective.

4 Conclusion

Sienna Williams, Dane Gleason, Sadhana Lolla, and Nick Dow

In Lab 3, we got our hands on real robotic hardware, transferred our simulation wall-follower to the racecar, and added a safety controller function that safeguards the car from collisions. This lab is the first step towards our goal of a fully autonomous racecar that can participate in the final challenge of racing around a closed loop track. The wall-follower capability will allow us to navigate confined spaces and park in front of objects as a part of future labs while the safety controller will continue to be necessary to ensure the integrity of the robot.

The racecar's wall-follower behavior was based on our wall-follower simulation code that was developed in Lab 2. Thus, much of the design and development of the wall-following system was already complete and only minor tweaks were necessary. Adjustments to the PID control gains as well as modification to the data output by the lidar scanner allowed our simulation code to work

on the actual robot. The combination of the PID controller and the front wall detector allowed our racecar to be able to follow straight walls as well properly turn around acute and obtuse corners. An experimental evaluation was conducted on the wall-follower by placing the racecar near various kinds of walls on both the left and the right side at various speeds. Analysis was then conducted on the collected data to convey that the wall-following behavior met technical requirements.

Each racecar is extremely expensive; for example, the lidar alone costs over $4,000. The safety controller was implemented to protect the components and prevent collisions with walls and other objects. The safety controller needed to be robust to protect the equipment, but also not so conservative that it would prevent the normal functionality of the racecar. The Velodyne lidar put a constraint on the minimum distance that the robot is able to see, so our safety controller had to stop within 0.5m from any objects to absolve this uncertainty. We tested the safety controller by running the wall-follower algorithm such that the racecar would be forced to run into a wall in front of it; we would then observe whether the safety controller successfully braked the car. We found that our safety controller stopped at the desired distance in every test, but that it was necessarily overprotective since we could not use a distance smaller than 0.5m.

The result of this lab is a successful wall-following and safety controller system that works on a wide variety of parameters and environments, including high speeds, gaps in walls, and corners. Although we were successful in implementing the technical requirements of the lab, there is room for improvement and discussion on open challenges. Some of these areas include improving lidar data parsing and slicing, more rigorous wall-following PID tuning, and modifying the safety controller braking algorithm to better address lidar sensor limitations in order to improve performance while maintaining safety.

5 Lessons Learned

5.1 Dane Gleason

I learned that with implemented software, real-world testing with hardware is a powerful and effective debugging tool with sensors interfacing with the world around them and that diverse testing scenarios are especially important. With this, I also found that hardware is often difficult and successful implementation is not as easy as running a simulation. Regarding communication, the biggest challenge in this lab was finding mutual availability for work sessions and the balance of work distribution. It was especially difficult with nominal task assignment at the outset of the lab, to then confront realities of schedules and bottlenecks in which someone else completes a task instead. The most im-

portant takeaway was the need to communicate quickly and clearly about the division of tasks, bandwidth, and availability for effective team work, especially considering the intensity of this class. Going forward I hope to utilize the car as much as possible as a debugging tool and to facilitate good communication practices to maximize our productivity and efficiency as a team.

5.2 Sienna Williams

I learned that not everything can be fixed to perfection and sometimes you have to make do with what you have and just explain your shortcomings. We learned that our sensor does not detect distances closer than 0.5m and instead just outputs infinity as the distance. We thought that we could get this fixed somehow and be able to detect closer distances, but after talking to the teaching staff we learned that this was a shortcoming we would just have to deal with and explain. In order to get around this problem, we did all our tests at distances greater than 0.5m so that our sensor would properly sense all the distances to the walls. I also learned that it can be difficult to find times that everyone is available to work. To get around this problem, we had some people work on the report when they could not work on the robot in person and vice versa to ensure that all the work got done.

5.3 Nick Dow

I learned that often problems need multiple perspectives to resolve issues, especially in debugging tasks where people often forget what assumptions they are making about the hardware/software they are working with. This phenomenon was most obvious when some team-member had been debugging the robot for a couple hours and another member would join, ask a couple unconsidered questions, we would investigate the issues those questions posed, and often then quickly resolve the issue. Also, I learned that perfection is the enemy of good, and that one cannot consider all possible outcomes at the outset of creating an autonomous system; the best course of action is often the quickest to a working prototype that you can then iterate upon. Communication was also a pain point in the lab, as we set out general tasks and goals for each team member to be responsible for early on, but did not establish structured communications for accountability.

5.4 Jensen Curtiss

Life can be painful at times, and this week was an example of that in my life for reasons outside of classwork. Despite this, I learned that communication is critical for all members to successfully complete anything, and I will admit that I struggled with communicating my hardship out of a misplaced belief that I could simply power through it and not have it bring anyone down. Powering

through things leads to burning out. Just like with the racecar as we discovered while testing, if you push yourself for too long without breaks, your system (my body in this case) will plan a break for you. With all this in mind, I intend to help our team structure a more effective communication system with a way of keeping track of tasks instead of nebulous messages that are lost in the flow of the project.