Lab 3 Report: Wall Follower

Team 16

Lasya Balachandran
Fiona Gillespie
Adrian Gutierrez
Jason Li
Katherine Lin

Robotics: Science and Systems

March 11, 2023

1 Introduction *(Adrian)*

Our group aimed to develop a race car that autonomously navigates its surroundings while avoiding collisions that could damage itself and its surroundings. To this end, we recently developed a program that will allow the race car to drive by following a wall on its left or right. This also allows the race car to navigate corners. In addition, we also implemented a safety controller to help the car avoid collisions. This program builds off the previous week of testing our different implementations in simulation and learning Linux, Git, and ROS.

The purpose of this lab was to use our wall follower implementation from the previous week and transfer it to a physical robot. Then we had to fine-tune it as needed for it to work outside of the simulation. The specifications of this lab were to log on and manually drive the car and visualize the laser scan. Then use our wall follower program to autonomously drive the race car and to use our safety controller to prevent crashes.

The problem for this lab was to make sure our robot appropriately handled inputs from the LiDAR to adjust its steering commands so that it maintained a specified desired distance from the wall. Then at any point when the racecar was in a position where it could not move forward without crashing, it stopped.

Our technical solution involved retrieving the laser scan from our LiDAR and fitting a line to that data to give our racecar a clean line to maintain a desired distance from. From here we used PID control to adjust our car's steering angle to accomplish maintain our desired distance from the wall. For our safety controller, we retrieved our car's speed and distance from the wall and based on our speed and distance we would stop the car if it was approaching a wall. In other words, the car would stop sooner if it was moving at high speed and later if it was moving at slower speeds.

2 Technical Approach *(Lasya, Fiona)*

In order to create a robot that could safely and successfully follow the wall, we focused on three main aspects: visualizing laser scan data from the LiDAR, adapting our wall follower simulation to the hardware, and creating a safety controller to avoid collisions.

2.1 Visualizing the Laser Scan *(Lasya)*

The robot contained a LiDAR sensor, which allowed it to view a portion of surrounding objects as a set of points. A visualization of these points can be seen in Figure 1. These points are read as LaserScan messages, which contain a list of distances from the robot to each laser scan point.

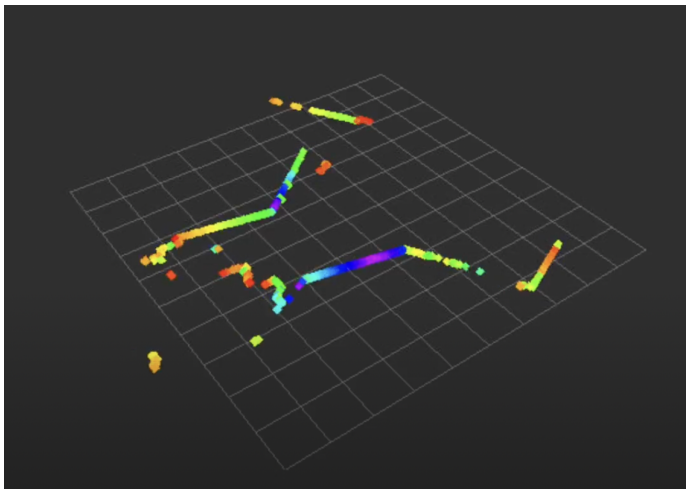


Figure 1: RViz of Laser Scan Data for Racecar in Stata Basement

2.2 Wall Follower *(Lasya)*

We created a wall follower with five main components: detecting laser scan points, implementing a corner detection system, filtering out unnecessary LiDAR data, using a least squares regression line to account for noise and outliers, and implementing PID control to adjust the error in the distance from the wall and adapt to corners and curves in the wall. Figure 2 shows a block diagram of our wall follower system.

Figure 2: Block Diagram of Wall Follower System

2.2.1 Corner Detection and Filtering LiDAR Data *(Lasya)*

The wall follower received a LaserScan object representing LiDAR data as an input. This data included a list of distances from the robot to each laser scan point where the indices of the list corresponded to angles running from a minimum angle from the robot to a maximum angle, as shown in Figure 3. For filtering, the data was split in three equal parts: right, forward, and left.

We then determined if the robot was at a corner to filter the data accordingly. If the robot was not at a corner, the robot only required the right data to follow the right wall and the left data to follow the left wall.

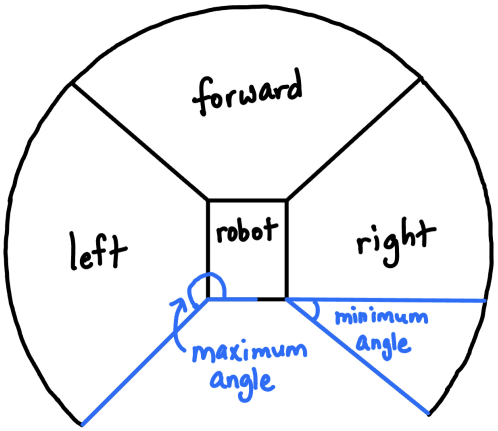
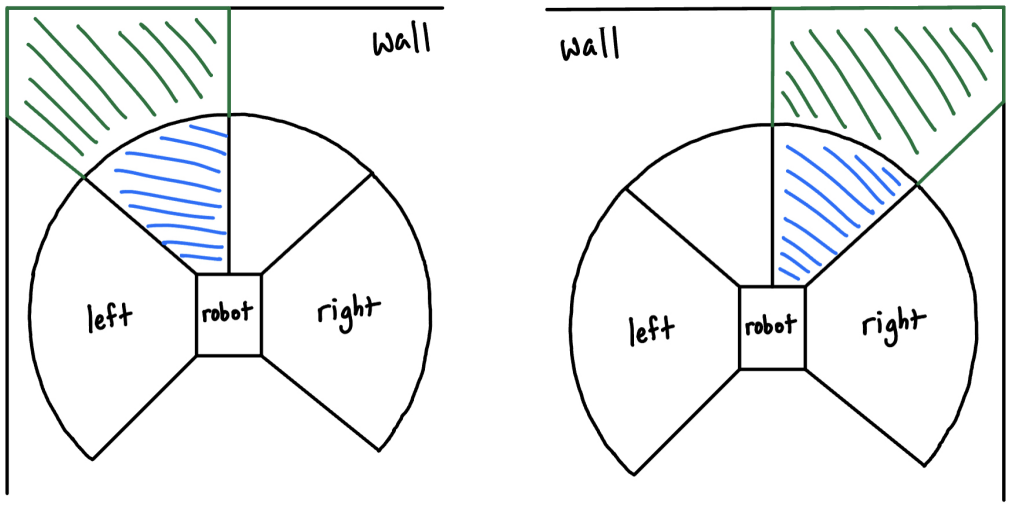


Figure 3: Right, Forward, and Right Sections of Data Based on Angles

However, when the robot approached a corner, it also had to take into account the points in the forward section. As shown in Figure 4(a), if the robot followed the left wall, it considered the points in the left half of the forward section, which eliminated noise compared to using the entire forward section. The robot detected a corner if the average distance of the 15 closest points in this section was less than 2.8 times the desired distance since we found this threshold to allow the robot to have enough time to find the corner and turn before reaching the wall. Similarly, when the robot followed the right wall, we used the average distance of the 15 closest points in the right half of the forward section, as shown in Figure 4(b). If the robot detected a corner, we used the left half (i.e. left and left half of forward sections) of the data for left wall following and the right half for right wall following.



(a) Left Wall Following

(b) Right Wall Following

Figure 4: Angles of Data Used for Corner Detection

After the data was filtered based on the right, forward, and left sections, we filtered out all points greater than 4 times the desired distance since those points would not affect the immediate trajectory of the robot. Once all of the data was filtered, we used the distances and angles of the laser scan points to convert the points to Cartesian coordinates, which were used to design a least squares regression line to minimize noise.

Based on the line, we designed a new set of points where the point closest to the robot represented the location of the wall.

2.2.2 PID Control *(Lasya)*

Our PID controller was then used to determine the steering angle of the robot based on the error between the desired distance and the distance of the robot to the wall. Figure 5 represents a block diagram of our PID controller for the robot.

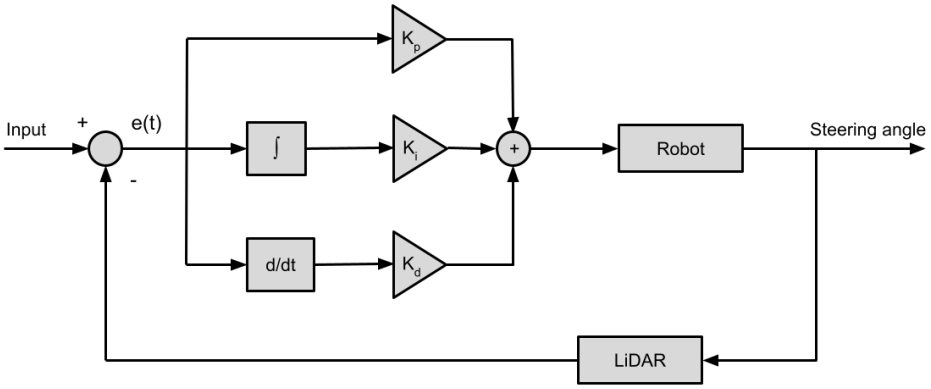


Figure 5: Block Diagram for PID Controller

When not at a corner, the minimum distance from the robot to a point on the least squares regression line represented the input for our PID control. However, this input did not work for corners since the set of points closest to the robot was on the right or left side until the robot was within a desired distance of the wall, which was too late for the robot to successfully turn around the corner. In order to solve this problem, we offset this minimum distance by the desired distance. As a result, the robot believed the wall was closer and turned sooner.

The error $e(t)$ was computed as the difference of the desired distance and the input. We used Equation 1 to compute the steering angle of car based on this error.

$$\text{steering angle} = K_p \cdot e(t) + K_i \cdot \int e(t)dt + K_d \cdot \frac{de}{dt}, \tag{1}$$
$$K_p = 0.4, K_i = 2.0, K_d = 0.4$$

where K_p was the proportional gain, K_i was the integral gain, and K_d was the derivative gain. The output was multiplied by -1 if the robot was following the left wall since the robot needed to turn in the opposite direction based on its distance from the wall.

2.2.3 Adaptations from Simulation to Robot *(Lasya)*

We initially designed our wall follower in simulation, and our robot generally worked well translating from the simulation to hardware. However, we ran into two main issues in adapting our design to hardware: oscillations in the robot while following the wall and turning corners.

To correct the oscillations, we changed the PID gains from $K_p = 1.0$, $K_i = 0.035$, and $K_d = 0.205$ to 0.4, 2.0 and 0.4, respectively. The lower K_p and higher K_d dampened the oscillations, while the higher K_i allowed the robot to eliminate error over time and remain susceptible to changes in the curves of the wall.

Our other issue was that the robot could not reliably turn the corners. While tuning the PID gains improved this feature, it did not fix the problem. Since the robot seemed to turn too late, we increased the threshold at which the robot detected a corner from 2.02 times the desired distance to 2.8 times the desired distance. In addition, we originally subtracted 0.5 for the PID input at a corner but found this to be an issue when testing desired distances significantly higher and lower than this value. Instead, we decided to subtract the desired distance from the minimum distance as the input for the PID controller, allowing the robot enough time to turn the corner while not veering too far from the wall.

2.3 Safety Controller *(Fiona)*

Our goal for the safety controller is to prevent crashes, either with static or moving objects in front of the robot. The safety controller will be running at all times, so it was important for us to create a careful balance between stopping the racecar for safety, without impeding the ultimate goals and tasks that the racecar is trying to complete (in this case, following walls).

2.3.1 Initial Approach to Safety Controller *(Fiona)*

In our initial approach, we first wanted to know more about how the racecar behaved. We needed to know how far the racecar would continue to move after we told it to stop. The distance that the car traveled after being told to stop is called the *stopping distance*. To determine this, we ran an experiment in the Stata basement where we had the racecar driving forward for a few seconds to reach a given velocity, then told it to stop after the racecar passed a certain point on the wall. We then measured the distance that the racecar travelled from that point. We used a ruler to measure this distance. Based on this experiment, we created an initial linear relationship between speed and stopping distance, as seen in Figure 6.

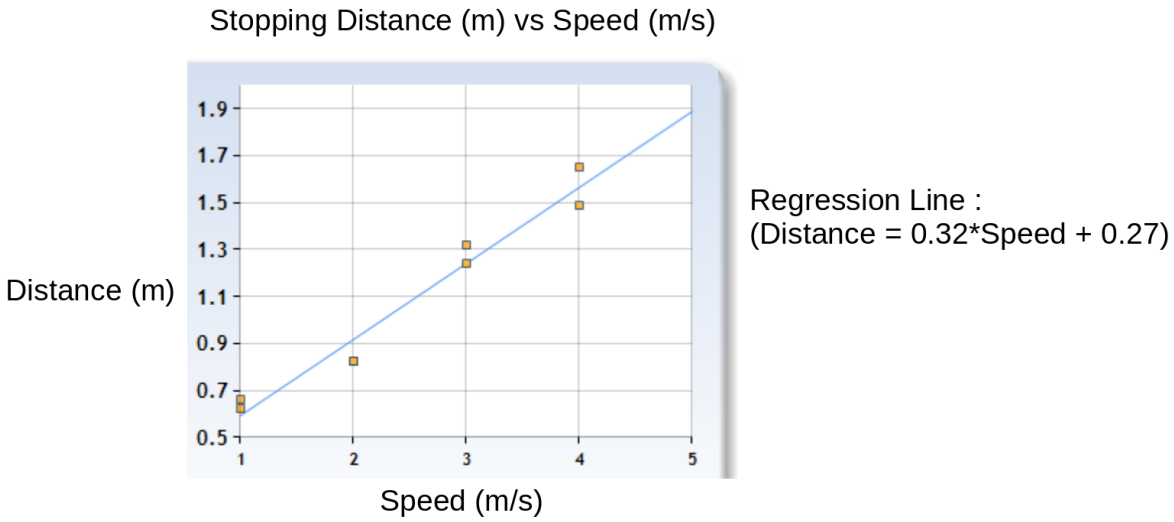


Figure 6: Linear Regression of stopping distance vs speed

2.3.2 Tuning the Safety Controller *(Fiona)*

Our experiment was helpful in creating a first step towards a safety controller. However, we quickly realized that our initial slope and intercept from the regression did not quite provide enough stopping distance during more rigorous testing with the wall follower. Our final safety controller check is displayed in Equation 2.

$$0.63 \cdot speed + 0.405 > \text{front distance} \tag{2}$$

This means that if the car is travelling at 2 m/s, it will stop if anything is closer than 2*0.63+0.405 = 1.665m away from the front of the car.

2.3.3 Handling Corners *(Fiona)*

The addition of the safety controller created a tradeoff between safety and accomplishing a task. Our safety controller initially made the racecar afraid of corners. Given our conservative stopping distance, the racecar would often stop as it approached a wall that it should turn at. To help moderate this issue, we added a corner sensitivity parameter that would allow the racecar a little more leeway in stopping distance if it was at a corner and already turning. This parameter was also tuned through testing, and ended up allowing an 80% stopping distance if at a corner, instead of the normal 100% of stopping distance.

2.4 ROS Implementation *(Fiona)*

For this lab we had two main ROS packages: a wall follower and a safety controller. We created a new ROS package for the safety controller for separation of concerns. The safety controller package takes in the LiDAR data and checked the forward section in the same way that the wall follower does, as outlined in Section 2.2.1: Corner Detection and Filtering LiDAR data. As a recap, we look at the front third of the scan data, and then only the half of that data which is opposite to the wall. If the robot was following the right wall, then we considered the left half of the front third of the scan. This is because some of wall from the side was interfering with our estimation of the closest object in front of the racecar. We decided to compute the forward distance in the safety controller independently so that the racecar would always be aware of its surroundings, even if the wall follower package went down unexpectedly.

The wall follower took in the scan data and outputted a drive command. The safety controller also took in the scan data, and can override the drive command by publishing to a topic of higher importance. Teleop superseded both of these for maximal safety. We also had the wall follower publish the percent error as a topic `\error` which we used in the evaluation portion.

3 Experimental Evaluation *(Jason, Katherine)*

We evaluated both the wall follower and safety controller functions of our robot through quantitative and qualitative metrics. While the integration between the wall follower and safety controller was successful and was observed to work many times, we chose to evaluate the wall follower and the safety controller separately.

3.1 Wall Follower *(Katherine)*

Just as in simulation, the goal of our wall follower on the real robot is to drive along a wall while maintaining a constant distance from it. Our focus when building our algorithm was creating one that was consistent, versatile, and adaptable. To achieve this, we used metrics that looked at and varied many parameters and evaluated the algorithm both qualitatively and quantitatively by measuring the average distance from the wall over the course of a trial. It was also important to ensure that the addition of the safety controller did not interfere with the robot's wall-following abilities.

We visualized the distance of the robot from the wall in real-time while varying starting distances, velocities, and which wall the robot was following. For all of these cases, it maintained a stable average distance from the wall with small oscillations around the desired distance. It should be noted that all observations and data included in this section were recorded with the safety controller also running on the robot.

Our metric, percent error, is calculated as $\frac{\text{(distance from wall)-(desired distance)}}{\text{desired distance}} * 100$. In Figure 7, we show three different cars set up close, medium, and far from the wall. We then ran our wall follower with desired distance set to 0.35 meters, following the right wall at velocity 1 m/s. As visualized in the graph, our robot did very well regardless of distance from wall; its oscillations consistently stayed below even 30% error once the car

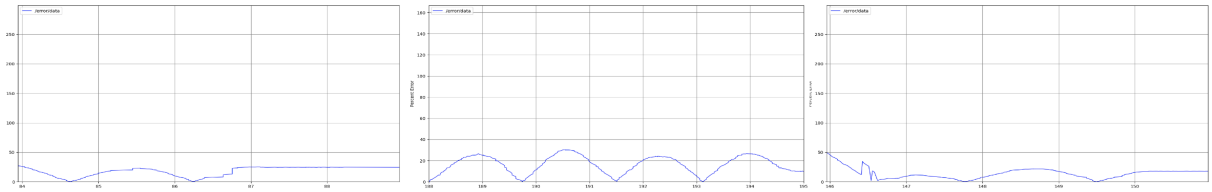


Figure 7: From left to right, these cars were set up 14 cm, 35 cm, and 105 cm from the wall. The error has been captured for each one when moving at velocity 1 m/s, following the right wall in a straight line.

got into a stable equilibrium with the wall. It also reached this equilibrium very fast in all three cases. This was the same for the left side as well, although that has not been visualized here to prevent repetition.

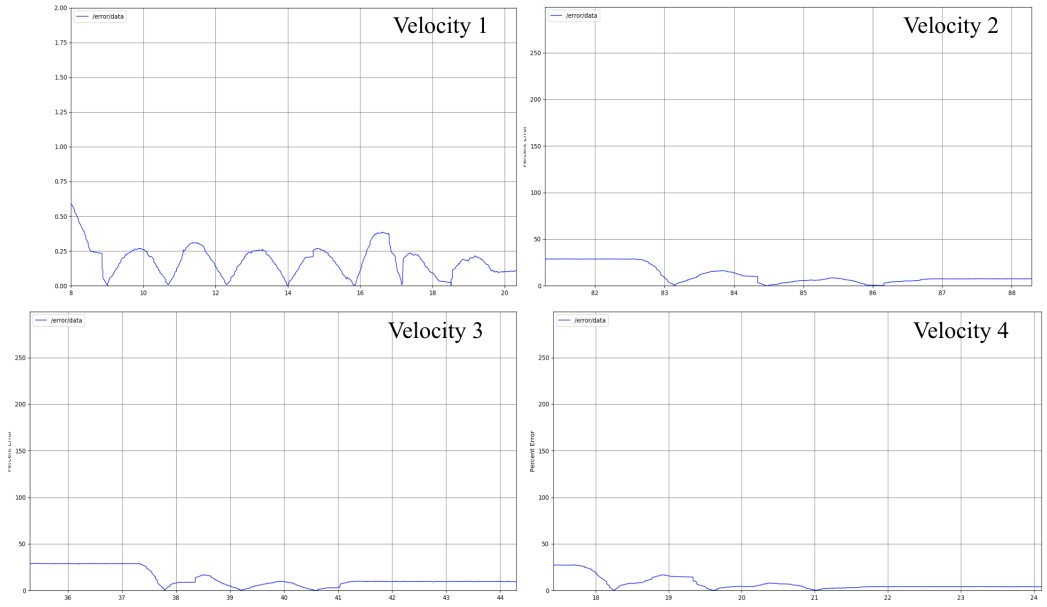


Figure 8: Each graph represents the error of a car moving at the velocity labeled in a straight line, following 35 cm from the left wall.

We also varied velocity. As shown in Figure 8, every velocity has a very low error, and at higher velocities, the car manages to smooth out its trajectory to create an even lower error, demonstrating the algorithm's adaptability.

Finally, we tested the robot on some corners. Given a complex series of corners to navigate, our algorithm completed the task successfully. We tested multiple corners in and around Stata and Building 26; Figure 9 is the error of the robot on the corners right outside of room 32-044. There are clearly many more oscillations than for a straight wall, with error even reaching 100% at one point. However, due to the higher difficulty of this task, the decline in performance is expected; this does not discount the success of the robot qualitatively in following corners.

The diversity of circumstances under which our algorithm has been shown to perform well indicates the versatility and consistency of our wall follower.

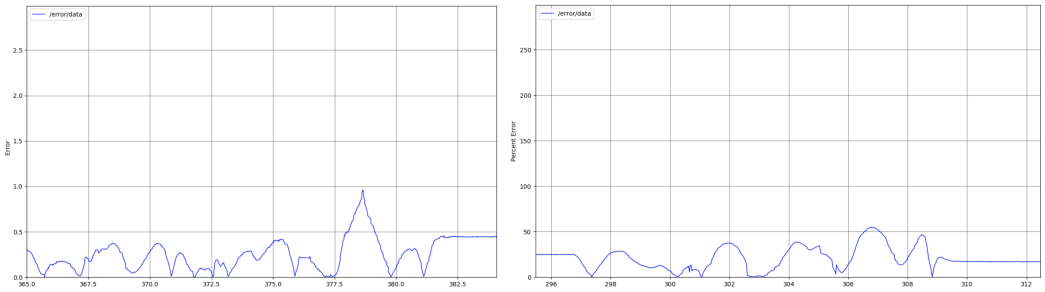


Figure 9: On the left is the error of the robot when following the left wall on a series of corners. On the right is the error of the robot when following the right wall on the same series of corners

3.2 Safety Controller *(Jason)*

Our safety controller successfully achieves its goal of preventing high-speed crashes that would be dangerous to the robot (or its surroundings) by stopping before impact with an obstacle in front of it. We made qualitative observations of the robot's stopping behavior in different experimental conditions and made quantitative measurements of how close it will approach an obstacle before stopping (which we term "obstacle distance") under a variety of speeds and obstacle types.

We tested our safety controller with 2 types of obstacles: stationary and moving. First, we intentionally tuned the wall follower's PID parameters by decreasing P until that the robot does not turn sharply enough to clear right angle inside corners without crashing into the forward wall; this served as our first obstacle type, an unavoidable, stationary obstacle. Second, we ran the wall follower with well-tuned parameters and quickly slid a cardboard box in front of the robot until completely intercepting its path; this represents our second obstacle type, an obstacle that suddenly moves to block the robot's path forwards (and then remains stationary). These two obstacle types should cover the vast majority of cases where the safety controller must be activated to stop the robot.

Using stationary obstacles (i.e., walls), we calibrated the safety controller's quantitative stopping thresholds under a variety of car speeds (please see Section 2.3.1 for detailed procedure and numerical figures). Qualitatively, we ensured that across multiple trials of each speed, the robot stops before making physical contact with the wall in front of it. This iterative tuning process resulted in a safety controller that consistently and robustly stops (at least 6 cm) before crashing into a stationary obstacle in front of it.

In Figure 10, we show, as a function of speed, the robot's obstacle distance, or minimum distance from a moving obstacle when it is stopped by the safety controller during wall-following. We chose to manually measure this distance in physical space using a ruler instead of computing it programmatically. Although this choice may sacrifice some precision, it avoids possible noise and error in the robot's *belief* about how far it stopped from the obstacle, and practically, we care most about how well the safety controller works in *real* space.

Figure 10 reveals that our car's minimum obstacle distance occurs around 5 cm, and that obstacle distance increases gently with car speed. This trend makes sense and is desirable, since a fast-moving car is more likely to skid or otherwise lose control, which warrants leaving a larger safety gap to prevent crashes. Note that, although we did not collect extensive numerical data, we observed that the robot also stops appropriately (at approximately the same range of distances) if the moving obstacle is a human who steps or "falls" in front of the car.

Therefore, across a variety of experimental conditions, our robot's safety controller is consistently able to stop forward motion to prevent crashes, always leaving at least 5 cm between itself and an obstacle.

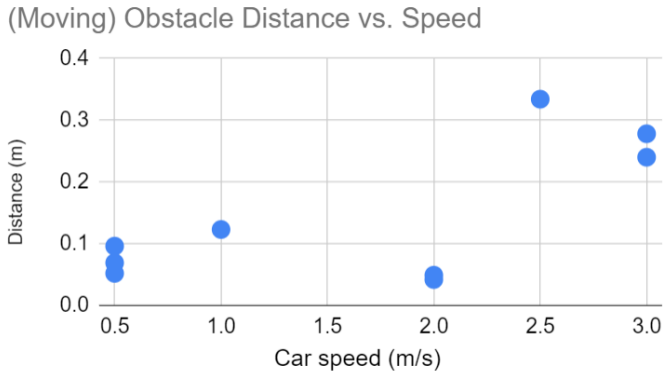


Figure 10: The robot's obstacle distance from a moving obstacle, defined as the minimum distance between any point on the robot and any point on the obstacle, as a function of the robot's speed setting during wall-following.

4 Conclusion *(Adrian)*

In this lab, we have been able to design a controller for a racecar robot that is able to follow a wall and turn corners. We were also able to jointly design a safety controller for this algorithm, which helped the robot avoid collisions that might otherwise cause damage to itself or its surroundings. This safety controller worked by stopping the car if an obstacle was detected in front. Though our algorithms were not without flaws, our robot was able to perform both wall following and safety braking successfully, as evaluated through qualitative and quantitative means.

In the future, we hope to be able to better tune our PID controls so that our racecar can drive with less oscillation than it currently exhibits. This would also likely help with future tasks; in general, less oscillation suggests smoother navigation and fewer unexpected orientations to debug. We may also take a closer look at our safety controller and make sure it can run in even more scenarios than we have currently tested; for instance, what might be the effect of angle of approach on stopping distance? Are there certain obstacle sizes or shapes that the safety controller does not work as well on?

However, no robot can be advanced without additional algorithms. We hope in the future to improve what we noted above but also take advantage of additional sensing modalities, such as image segmentation with video cameras, that the robots are equipped with to autonomously operate our robots in more robust scenarios that will imitate real world use cases.

5 Lessons Learned

5.1 Lasya

Since this was our first lab using hardware, I learned a lot about integrating the software with the hardware. It was interesting to see that our simulation code did not work nearly as well on our the hardware. While our code for this lab ended up remaining very similar to the original simulation code, this lab was still technically challenging in the sense that we had to figure out how to work around initial issues to teleop into the robot and later ran into some issues with the robot stopping every few seconds from time to time during tests and a broken battery charger.

For communication, I found the aspect of managing time to share the robots with other teams to be difficult. I found that this team worked together well from the beginning, and we communicated well with each other about the technical aspects and availability/division of work. However, for all of the teams in the pod, prioritizing a working wall follower and safety controller sometimes meant forgetting about the limited time with the robot. The limited time also led to teams having to negotiate with each other. However, as the week progressed, all of us in the pod were able to better communicate with each other about handoffs and slots with the robot.

In terms of collaboration, this lab was a bit difficult to divide up the work in that it involved adapting one person's wall follower simulation code to work on the hardware and creating a safety controller. We initially found it hard to divide up the work in this way but after the first few days, we found a solution in allowing different people to work on the code, control the joystick, and document the tests through videos and notes each day. This allowed all of us to gain experience in each area going forward.

5.2 Fiona

As this was our first lab with both our teams and the real hardware, I learned a lot on both fronts. In terms of communication and collaboration, I learned that it will be important moving forward to plan ahead for how we can parallelize the work. For this lab it was hard to have all 5 of us actively working at times, but I think the future labs also have a structure that lends itself very well to parallelization. In terms of best software practices, I learned it is very important to set up our github so we can easily push and pull from github to a local computer to the car. In terms of the hardware, I learned there are lots of new things to check - from battery levels, to moving files between the car and a laptop, to considering how the real environment might be affecting the performance of the car in new ways (such as different buildings have floors with different friction levels). Overall, I've learned a lot in this lab and I look forward to learning more with my team!

5.3 Adrian

Though I had already been familiar with some of the topics mentioned up to now it was a lot more complex to implement the desired algorithms myself. Though interesting, there were a lot of issues showing that theoretical results don't always translate 1:1 to real life. It was a lot of trial and error and trying to decide if an issue was an internet, hardware, or actual programming problem. Nonetheless, it did get easier throughout the week as I became more familiar with the robot and the packages we utilize and I presume that this will continue throughout the semester.

On the communication side, I felt that it flowed a lot smoother than I had thought it would. Within my own team, it was a bit difficult at the start given that no one knew each other but I feel that at this point we have been able to foster a collaborative setting. With the other pods, it does seem a lot more difficult. Organizing when and where we can hand off the robots and allocating sufficient robot time for each team did get finicky but I believe that each team has been able to at least promptly respond and try to figure out solutions to these problems even if the end result wasn't perfect.

Collaboration can also be difficult. When one person is running and editing the program on their computer and another is just using the joystick it can feel a bit unbalanced. Ultimately my own program was not chosen to be implemented on the robot so there was a bit of a disconnect between how the code we did use actually worked even after we were given a high-level overview of it. Though ultimately it worked out as I did eventually get to run some commands on my own and get familiar with the program. We also each took turns on different tasks so everyone had a sufficient understanding of how to use the robot. I believe being able to rotate these tasks around (running the program, using the joystick, editing the program, etc) did allow us all to equally contribute to our solution and be able to bounce ideas off each other for any problems that we might encounter. Ultimately, I believe, that in the following weeks, this too will improve.

5.4 Jason

Foremost among the technical lessons that this lab taught me was the difficult reality of translating theory and software into real hardware. Even an algorithm that works very well in simulation may not work nearly as well in practice due to a variety of factors that are difficult to control for in the real world, such as the environment and the robot's construction itself. Additionally, due to the many pieces of complex interacting components that comprise a robot's hardware and software, making effective and efficient use of ROS is no trivial task and requires good organization and a clear mind to sort through all the layers of nodes and topics, not to mention the 10 terminal screens open at any given time.

Communication- and collaboration-wise, I learned the importance of organization, both in terms of information and in terms of division of labor. Sometimes, our code files were scattered, with different people seeing different versions; more frequent reliance on Github as a central repository would help with this in the future. Additionally, with many team members and many different tests to run, it was important to keep track of all collected data in a central location that not only any one of us can access, but also clearly labeled so that any one of us can understand what was done and how it was done. This will become even more important given my final point, which is about efficiency and parallelization. While everyone contributed well to all parts of the lab, there was certainly a lot of time where several of us in the team were idling, not making the most efficient use of our time. For instance, instead of observing the experiments being run, some of us could have started on data analysis and visualization to speed up progress for everyone.

5.5 Katherine

This lab was my first time using hardware/a real robot, so I definitely had a lot to learn. From a technical perspective, I've learned a lot about the differences between simulation and hardware, and how bridging the gap between the two might require quite a bit more work than just using the same code. Throughout the past week, we experienced a multitude of issues with our robot, and it was very enlightening to see all the maintenance required to work with our robot. All of the many inconveniences I've encountered in simulation (having to rerun the code for every small change, restarting the RViz every time, etc.) became even greater in real life since we had to re-place the robot in the desired space and rerun multiple commands with every edit. I've also never given much thought to how robots are able to use our code, so I spent a lot of time learning about the ssh process, pushing code to the robot, and the general robot architecture.

From a collaborative standpoint, this is my first time working in such a large group for a task like this. It was definitely a challenge to find things to do sometimes since usually our work with the robots didn't require five people. However, I do think that later in the week we definitely learned to take advantage of this, since we could greatly modularize tasks like taking a rosbag while recording a video, while running multiple commands and editing the code, and teleoping the robot.

Finally, for the communication aspect, I felt that the Lincoln presentation given in class really emphasized the downsides of slides, and while making the presentation for this week's briefing, I got a chance to implement the techniques mentioned in class to mitigate the issue with slides (keeping slides free of text and making slide titles full sentences).