Lab 3 Report: Wall Following

Team 13

Alazar Lemma
Benjamin Soria
Caden Moore
Henry Heiberger
Vittal Thirumalai

Report edited by Benjamin Soria

Robotics: Science and Systems

March 11, 2023

1 Introduction

authored by Caden Moore

As NEET students in the autonomous machines thread, we have developed rudimentary robots that can follow lines on the ground using line sensors or maintain distance away from objects using basic distance detectors such as ultrasonic sensors for autonomous navigation. These were the simple methods we used to create robots that could traverse an environment autonomously in previous classes, but these sensors are so limited in their simplicity that our robots were consequently severely limited. In 16.405/6.4200, these limitations are thrown out of the window with the racecar platform and the only thing holding us back is our ability to implement code and algorithms. With that being said, with more advanced sensors comes more complexity, and more complexity almost always makes things more difficult. This report highlights the adversity we overcame and how we dealt with those difficulties.

In this lab, we were tasked with developing an algorithm for our racecar that can follow a wall and at the same time not crash into anything in front of it. We successfully developed the foundations of our autonomous racecar by implementing a basic wall-following algorithm that uses a multitude of lidar data points to identify a wall and maintain a desired distance away from said wall in addition to a safety controller that is used to avoid damaging our very expensive robot. These first steps are critical in the development of an autonomous racecar because it is necessary for our car to navigate through the world while avoiding obstacles, and both of the functions we implemented are the basics of achieving this. To navigate through the world our robot needs the ability to identify a path which is determined by the walls and wall following, and we want our robot to avoid collisions to avoid damaging the robot which is achieved using our safety controller.

These functions that we implemented fit well and are complemented by the previous labs we have completed. Starting with the intro to RSS and intro to Git labs was a huge help in developing our basic necessary skills to build the wall follower, and being able to test and hone in our wall following algorithms on a virtual robot in a simulation made the transition over to a real robot much easier than if we were to start developing from scratch. Not only would it have been much harder to start from scratch, but it would have been much more dangerous and we would risk damaging the robot in the process while debugging. Although we had these initial labs to help aid us in implementing wall following and a safety controller on our robot, as a group we all encountered many challenges along the way. We struggled to overcome many of them but we communicated effectively and efficiently with one another and were able to conquer our challenges with resilience. This report will guide one through the steps we took to create a high performance competitive autonomous racecar.

2 Technical Approach

2.1 Developing the Safety Controller

Authored by Henry Heiberger

Of the two primary technical problems we needed to solve for this lab, the first problem that we approached was the safety controller, a module intended to improve the safety of operating our racecar robot for the remainder of this lab and future labs. This section highlights and rationalizes our technical approach to our safety controller as well as discusses its implementation at depth

Specifically, the safety controller is a module whose main purpose is to prevent the car or any object or pedestrian that enters its path from getting injured or damaged. Given this, we designed our safety controller with the goal that, in the event that the robot is rapidly approaching an object head-on, the module would trigger, slowing the vehicle down enough or stopping it such that both itself and the impacted object wouldn't be affected. However, it is important to note that this safety controller is also a module that was operating throughout the second half of this lab and will be operating during all future labs. Thus, it was also important to us that the safety controller was designed to be flexible and not overprotective because stopping too liberally would hamper the robot's autonomous driving ability. This gave us two primary design goals when designing our safety controller: both **safety** and **flexibility**.

To approach these goals, we first needed a method to determine whether an object was in front of our racecar. To achieve this, we made the decision to rely on the car's LIDAR sensor, a clear choice due to its ability to quickly and accurately report distance values back to the vehicle's computer. As shown in **Figure 1**, specifically, we considered a narrow slice of LIDAR data that covered a cone spread 20 degrees in both directions from an axis extending out the front of the car and took the minimum of that data to determine how far the robot was from any potential collision. The exact bounds of the slice of LIDAR data we examined were chosen by experimenting with many different options and choosing the one that detected an area approximately the width of the vehicle chassis (as shown when examining a visualization of the LIDAR scan) when the car was approximately a meter away from the wall. This was chosen because we wanted to ensure flexibility by only detecting areas of the wall that were inside the path of the car. In addition, we used the minimum because we wanted to ensure safety by making sure that objects of most reasonable sizes in the path of the car would still trigger the safety controller, even if they didn't obstruct the entire LIDAR sample's field of view. Given that to actually override the drive command being issued to the robot and make it stop, the safety controller's stop command needs to be continually issued for many cycles of the controller, we further reasoned that our controller was not very susceptible to potential outliers or false LIDAR readings, making us not concerned about using the minimum reading for this controller.

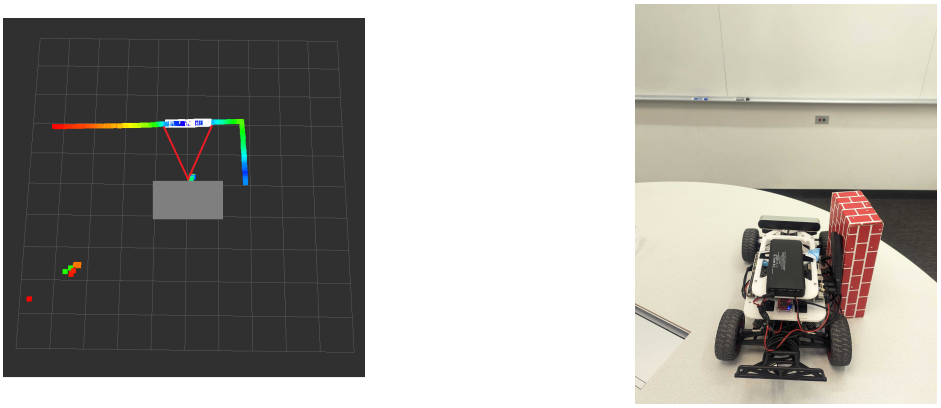


Figure 1: The left image contains annotated LIDAR data visualizing how the robot detects potential collisions in front of it. In it, the gray rectangle represents the racecar, the red lines represent an estimation of the angles in which the collision detection algorithm is sampling, the colored dots are LIDAR samples, and the white dots are the LIDAR samples that were automatically recognized as objects in front of the car. The right image shows the placement of the racecar when the image was taken, with the brick being used to make a LIDAR point appear at the location of the robot.

Beyond object detection, our safety controller needed to be able to decelerate the robot as quickly as possible. We reasoned that this increased safety by helping the robot stop even in tight situations and improved flexibility by allowing us to reduce how far the robot needed to be from the wall before it triggered a stop command. However, our robotic racecars did not have any built-in braking systems. Thus, to simulate this the best we could, our team opted to make the car slightly reverse whenever a stop command was issued. While a reverse velocity of -0.1 m/s was chosen such that the racecar would not actually move backward when a stop command was issued so that operating the safety controller would not affect the final position of the vehicle (instead only stop its movement), doing so provided an opposing force that hastened the deceleration of the racecar when compared experimentally to just issuing a stop command with a velocity of only 0 m/s. Given this, it felt like a reasonable choice for our team to implement this command into our final algorithm.

After finalizing object detection and the stopping procedure, the final design decision we made when designing our safety controller was when to stop the car. Though the idea of stopping the car is relatively straightforward if the robot is always operating at a constant velocity, one factor that complicates the development of an effective safety controller is the fact that the car's velocity when it's in a scenario where the safety controller could be needed is unpredictable. Thus, we concluded that any safety controller we designed must take the car's current velocity into account, ensuring safety by allowing us to confirm better that the safety controller is effective at all vehicle speeds and ensuring flexibility because it allowed us to optimize the distance where a stop command was issued to the speed that the vehicle was driving at (and not stop at all if the vehicle wasn't driving forward). Because our racecar doesn't have a direct way to measure its velocity, we made the decision to approximate it using the velocity of the last drive command issued to the vehicle. This was obtained by applying a ROS Subscriber that listened to the */vesc/high_level/ackermann_cmd_mux/output* topic (the ROS topic where all high-level drive commands are issued). This seemed reasonable because unless the robot was already rapidly decelerating (a case where the safety controller would be less critical), this drive command should provide a rough upper bound on the vehicle's velocity.

Once we had estimated our racecar's forward velocity using the last drive command issued to it, we experimentally determined the distance the car should issue its stop command when it was traveling at 1 m/s, 2 m/s, 3 m/s, and 4 m/s. These points were chosen such that when the safety controller issued a stop command when the car was at that distance, the car would repeatedly come to a stop $3-5$ inches from the wall (a small leeway to allow for varying motor speeds and ground friction). To then extrapolate this data for all velocities less than the car's maximum speed of 4 m/s, we applied the exponential regression that can be seen in **Figure 2**. This gave us a $y = AB^x$ model that we could plug into our safety controller and make it reasonably adjust how close it was willing to allow the racecar to

drive toward the wall before issuing a stop command based on the speed it was going. This change in the distance where the stop command is issued can be seen in **Figure 3**.

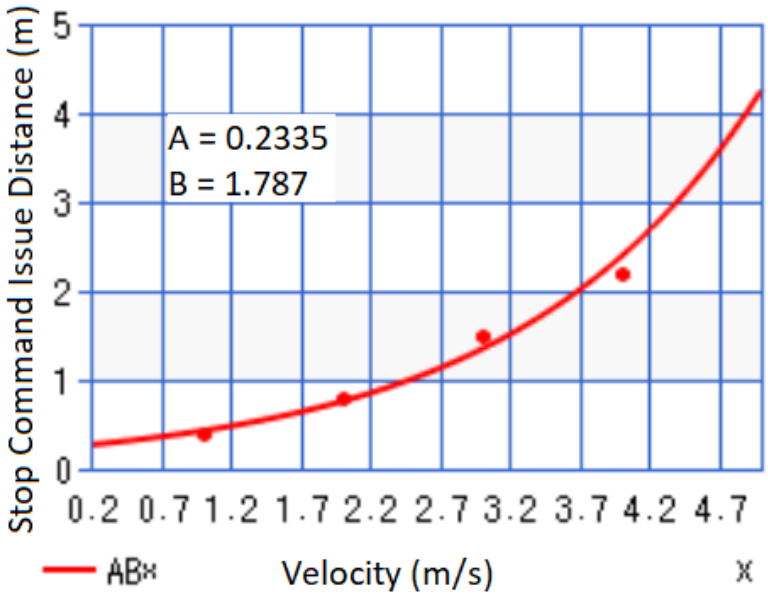


Figure 2: The four data points represent our experimentally determined distances in which the safety controller should issue stop commands to allow the racecar to come to a stop $3-5$ inches from the wall. An exponential regression was taken to extrapolate the data for all velocities less than 4 m/s.

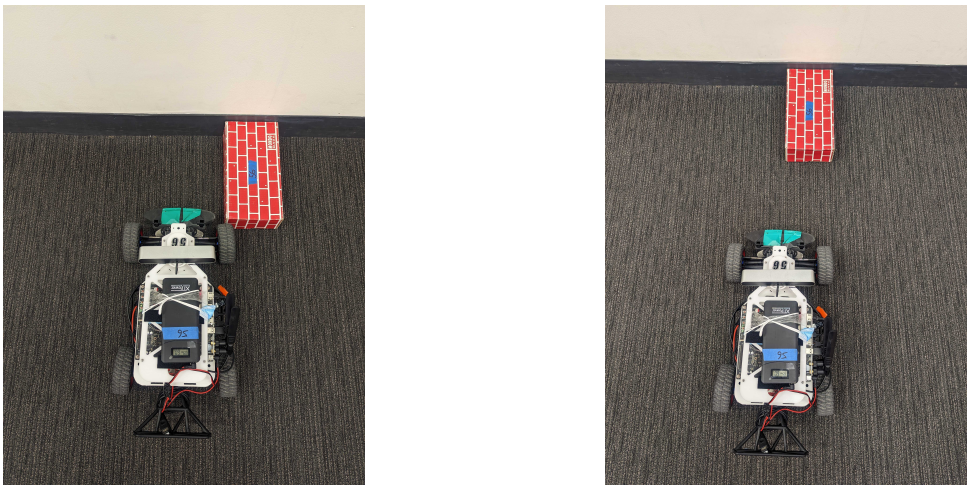


Figure 3: The left image shows the distance in which a stop command is issued by the safety controller when the car is traveling at 1 m/s. The right image shows the distance for when the car is traveling at 2 m/s, just under twice the distance of the 1 m/s case.

Putting these three components together, our safety controller protects the car and those in its path by using the minimum of a narrow slice of its LIDAR data to determine how far away the car is from potential collisions, applying an experimentally-determined exponential regression to decide how close it can safely get to the wall before issuing a stop command, and then, if necessary, triggering the racecar to stop by applying a slight negative velocity drive command.

2.2 Developing the Autonomous Wall Follower

Authored by Benjamin Soria

The second technical problem that we approached was the implementation of our autonomous wall-following algorithm, a highly critical section that would have future implications for the rest of the semester. This section features our decisions, our rationale for making the wall follower algorithm, and its implementation.

Wall-following algorithms enable robots to navigate a three-dimensional world by creating a two-dimensional representation. With the addition of constraining motion to a 'set' distance from the wall, the challenge is reduced to a control theory problem.

All members of the team individually created their own algorithms in simulation. Ideally, we would have tested everyone's algorithm to measure how well it performed in the real world compared to simulation. However, due to unanticipated difficulties in reserving the robot, establishing a collaborative GitHub repository, and scheduling team members' time, our team decided to shift our strategy. We discussed similarities and differences in the way we approached the simulation challenge and complemented this by showing how aspects of the algorithm affected the overall performance. We realized that all members sectioned the LIDAR data into three sections. A leftward view, a front view, and a right view. Four out of the five members applied linear regression to the LIDAR data. Of these four, two members removed outliers from the data prior to linear regression, either by removing data outside the interquartile range or a beyond a maximum threshold. One member implemented the RANSAC algorithm and accomplished both tasks simultaneously. The next fork in strategy concerned defining the robot's distance to the wall. Two members defined the observed distance to the wall as the orthogonal projection of the derived line onto the robot's origin; two members defined the observed distance as the y-intercept; one member used the equation of the line to measure distance as the y-value corresponding to one meter forward in the +x direction. Once a measured distance to the wall was found, all the algorithms converged to the same strategy: performing Proportional Integral Derivative control on the measured error.

It was clear from the simulation that the cars had to disregard outlier data from the LIDAR scan, the interquartile range strategy was chosen because it was robust to gaps in walls, such as open doors and obstructions. Taking the orthogonal projection of a line found by linear regression onto the car worked better than considering the y-intercept because the y-intercept changed dramatically depending on the car's orientation. This is to say, rotating the car about its' origin produced different observed distances to the wall. Taking the orthogonal projection of the wall from the car's future state also worked very well, but it amounted to considering the velocity of the car since this future state in the +x direction could be considered the position measured one second into the future.Since we anticipated implementing derivative control, this strategy would be incorporated into the control. The overall strategy developed as follows

- Collect and parse the laser scan data into sections
- Remove outliers and perform linear regression on the data
- Define distance to the wall as the orthogonal projection of the wall onto the robot
- Perform PID control on the error, defined as $e = -1(setpoint - distance)$

2.3 PID Control

Authored by Benjamin Soria

Given the equation of a line in a slope intercept form, our team calculated the observed distance to the wall, e using the model shown in Figure 4. For a wall located on the right side, the equation of the line could be written as $y = mx - b$ where $b > 0$, $m > 0$, and the following relationships could be derived.

$$\alpha = \frac{\Delta X}{\Delta Y} = \arctan(\frac{1}{-m}) \tag{1a}$$

$$e = b \times \cos(\alpha) \tag{1b}$$

$$\frac{de}{dt} = V_b = V \times \sin\alpha \tag{1c}$$

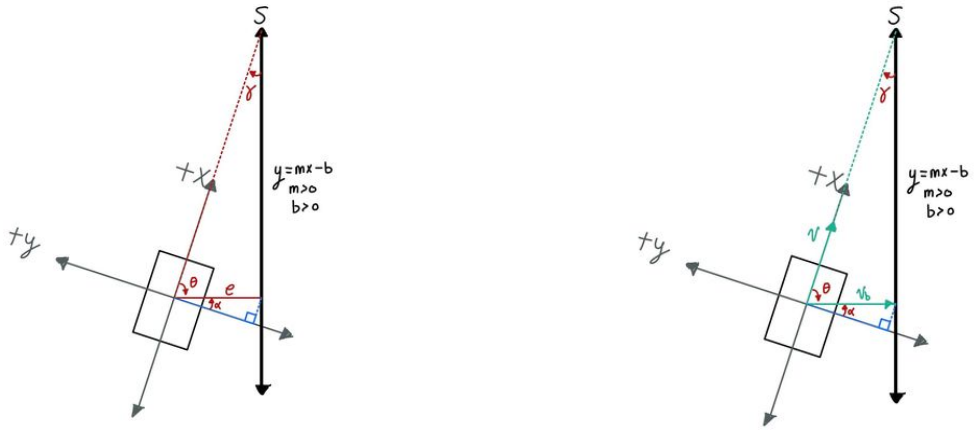


Figure 4: The model for our derived PID Control utilizes trigonometry to measure the distance to the wall. $y = mx - b$ characterizes a line fitted to the right wall, e is the distance to the wall, θ is the angle formed by the +x axis and the derived line, α is the ratio of the rise in x to rise in y direction given by the line. V is the forward velocity from the car's axis, V_b is the orthogonal component of the car's velocity to the line

This model ensured that the error E could be written as $E = -1(s - e)$, where s represents a setpoint. Then, k_p and k_d factor into our control C, the steering angle command, as follows.

$$P_{gain} = k_p \times E \tag{2a}$$

$$D_{gain} = k_d \times \frac{dE}{dt} = k_d \times \frac{de}{dt} \tag{2b}$$

$$C = P_{gain} + D_{gain} \tag{2c}$$

Our implementation results in a P_{gain} that seeks to minimize the error and a D_{gain} which accounts for the orientation of the car relative to the setpoint, summarized in Table 1. In scenarios where the car is already on a trajectory that will minimize the error, i.e the car is facing the setpoint, the D_{gain} reduces the control effort, since it will have a sign opposite the proportional gain, and vice versa for when the car is on a trajectory away from the setpoint.

Does +x axis intersects setpoint	P_{gain}	D_{gain}
Yes	Increases Control	Reduces Control
No	Increases Control	Increases Control

Table 1: PID Control Effort Summary

2.4 Corner Override Algorithm

Authored by Vittal Thirumalai

One problem with the PID control that we noticed through testing is that it worked well on a straight or curved wall, but it was delayed in response when turning at a corner. To optimize our wall following algorithm for this scenario, we added an algorithm to override the PID control when a corner is detected, shown visually in **Figure 5**. Similar to the safety controller, the algorithm uses laserscan data within 20 degrees of the straight line ahead of the robot. If the average of this data is within a threshold distance, the robot steering angle is overrided to be at the maximum turn angle of 0.34 radians. This threshold is set so that the robot is as close as possible to the desired distance during the turn.

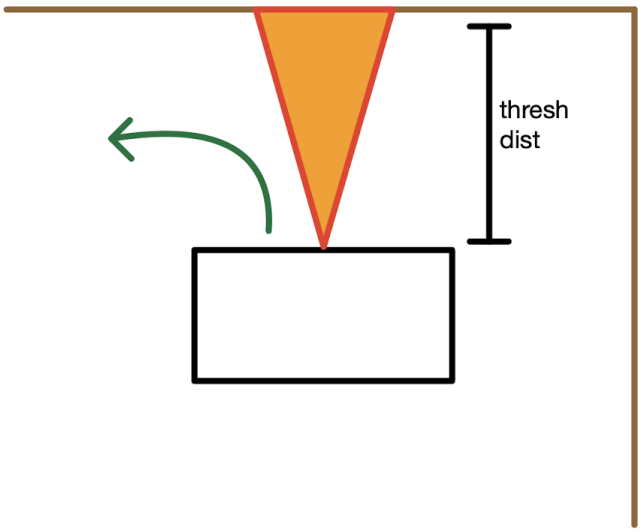


Figure 5: When the robot is within a threshold distance of the wall ahead, the corner algorithm overrides the PID control to make a sharp turn.

3 Experimental Evaluation

3.1 Wall Follower

Authored by Benjamin Soria

Lengthy discussion of the algorithm and its intended environment, clear roles defined for each subprocess, and rigorous testing of the algorithm in simulation enabled our team to assemble a working PID control wall following algorithm. All that was left to do was tune a series of parameters, kp, kd, lb, and ub to match the simulation performance in the real world, where kp and kd are the proportional and derivative gains, respectively, and lb and ub are the upper and lower bounds of the data we'd like to consider for linear regression. As can be seen in Figure 4, for a small angle α between the robot's forward trajectory and the wall, our error is approximately equal to the angle. We based the initial value of kp on this observation. For $kp = 1$ and along straight walls, the robot maintained a consistent one-meter distance to the wall. This was confirmed using a tape measure. Continuing with our testing, we attempted to have the robot follow a wall and make a turn at the first corner. Our robot failed this test at $kp = 1$, $kp = 2$ but succeeded at $kp = 3$. However, it didn't make sense to utilize $kp = 3$, since an error of 0.1 meters would result in a steering angle of 0.3 rad. A control operating in this manner would behave as a "bang-bang" controller, in essence switching between turning harshly left and turning harshly right in an attempt to oscillate over the setpoint. We realized we could keep kp small (thus commanding smaller steering angles) if the robot could somehow turn sooner.

This led to tuning the visualization window. Our LIDAR data is initially split into a left and right array. We process the array corresponding to the side we want to follow, converting polar data to cartesian form. The visualization window sets the fraction of the processed array to consider for linear regression. This procedure allows for future implementation of a varying visualization window corresponding to velocity or wall uniformity. We anticipate that at high speeds, we'd like to consider a larger fraction of the data, while at low speeds a smaller visualization window might correspond to a better linear regression fit. This feature also supports changes in visualization window due to irregular walls. When following jagged or noisy walls, such as a cliff or fence, we may want to smooth over the span of data by considering a larger visualization window. Ultimately, we slowly increased the visualization window from its initial range of $[\frac{1}{6}, \frac{2}{3}]$ of the right portion to $[\frac{1}{6}, 1]$ of the right portion, as shown in

Figure: 6. Though a fraction of this data corresponds to points directly in front of the robot, which we do not want in our linear regression, the outlier function removes these points prior to fitting. The points are slowly reintroduced into consideration for the best-fit line when they appear within the IQR of the entire side set, which enables the robot to perceive a smooth corner in place of a sudden corner.

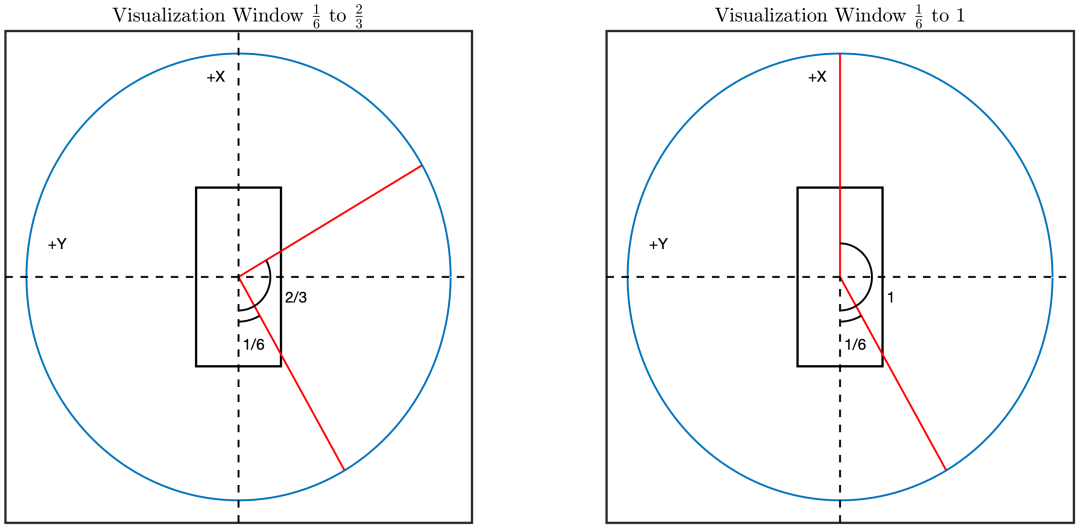
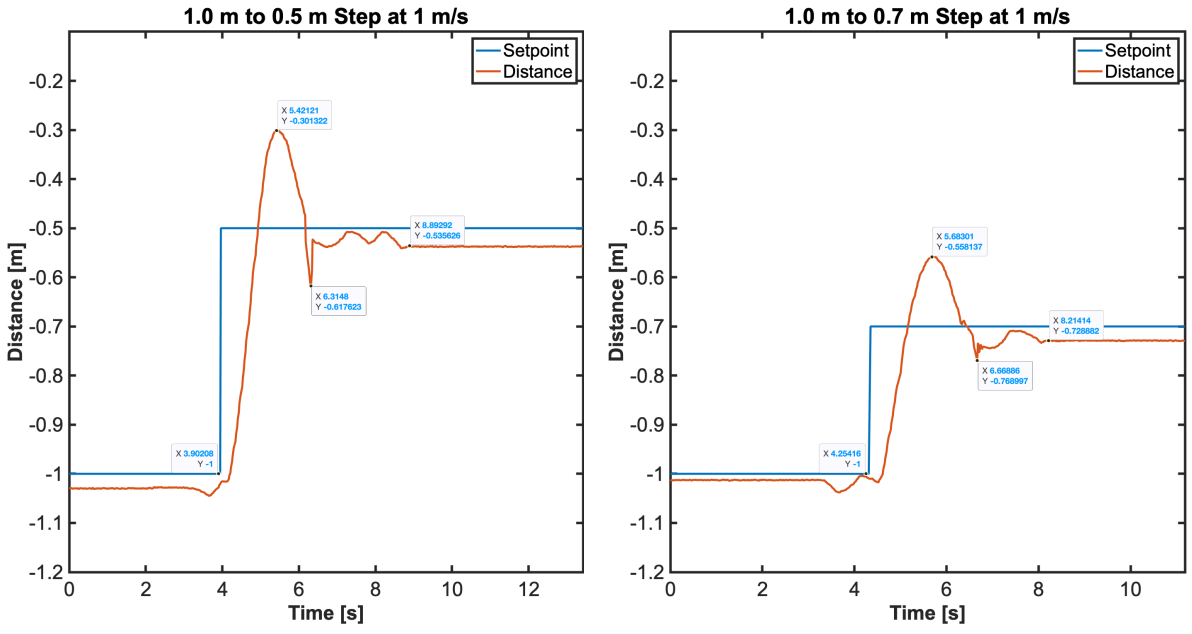


Figure 6: The visualization window splits LIDAR data into left and right portions. Using rospy params, we can select the fraction of left or right data to consider for linear regression.

Kd was the last parameter to be tuned. We tested $kd = 0.05$, $kd = 0.1$, and $kd = 0.2$. Beyond 0.2, kd overrode kp.For $kd = 0.05$ we achieved the most consistent wall following.



a) b)

Figure 7: Step Response for the right wall following. In both tests, the car traveled at 1 m/s. Steady-state values and peaks during transient are shown.

Figure 7 displays two step responses for the car traveling at one meter per second following the right wall. Tests were also conducted at 0.5 m/s and while following the left wall, but were omitted from the report

due to a lack of settling time data. In all cases, the steady state error never exceeded 0.05 m. Our team was extremely happy with this performance, since aiming, for zero steady-state error would ultimately require incorporating an integral term into our controller. The percent overshoot for plots a) and b) are 0.4% and 0.15%. Since our wall following algorithm removes outliers, applies linear regression to remove noise, and implements corner detection, it is unlikely we encounter step inputs while completing the final challenge. However, from the plots, we observe our racecar can respond to a 0.5-meter and 0.3-meter step with a settling time of about 8 seconds and 4 seconds, respectively. Additionally, it can be observed from the graphs, that the 5 percent settling time would be much lower that the steady state settling time.

3.2 Safety Controller

Authored by Alazar Lemma

Our group decided upon testing the reliability aspect of the safety controller, as we thought it would be prudent for it to stop safely and reliably, regardless of differing speeds. The safety controller is not built to stop the car a certain distance away, as that would take a proportional integral derivative controller at the least. This would be unnecessary given the purpose of the safety controller, which is to keep the car from receiving irreparable damage to itself.

To test the safety and reliability of the controller, we setup the car, and programmed it to drive straight into a wall at different speeds from 10 feet away. We then recorded how far from the wall it stopped before crashing using rosbag recordings, as well as the qualitative result of whether it crashed into the wall or not. MATLAB was used to process the rosbag data and transform it into data that we can utilize. For these tests, we repeated the tests 10 times for 1, 2, and 3 m/s, giving us a variable range of speeds that the PID controller could expose us to. Recording the distance from the wall when the car crashes allows us to record how confident we are the car will not crash given a certain speed.

We decided to not test speeds under 1 m/s because given the functionality of our safety controller, we predicted that the controller will work with speeds lower than that. Also, even if our controller were to fail with a speed less than 1, any collision damage would be negligible, if at all, which is in line with the goals of it being safe and reliable. It is at higher speeds where the integrity of the robot is at risk, so the safety controller is geared towards those speeds, and so it was at higher speeds where we decided to test. Even though the safety controller was designed with 4 m/s in mind, we decided to omit testing that speed because we thought that it would be very unlikely for the robot to ever reach a speed.

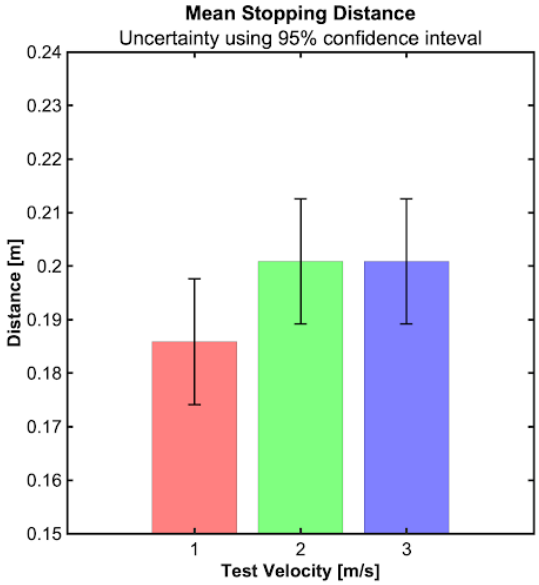


Figure 8: Results of Safety Controller tests. The error bars reflect the statistical probability that car veers off the expected stopping point.

As shown by the bar graph above, we can prove that our car is reliable given speeds of 1, 2, and 3 m/s, with a confidence interval of 95%. The minimum stopping distance of the car is recording a minimum LIDAR sensor distance of 0.2m from the wall. Qualitatively, we saw that the car did not crash into the wall at speeds of 1 and 2 m/s, but did slightly bump into the wall at 3 m/s. Based on the data and observations, we can safely assume that if that is the minimum distance the sensor is recording, that the car is either stopping right before the wall or colliding with such gentle force that the car would not take severe damage.

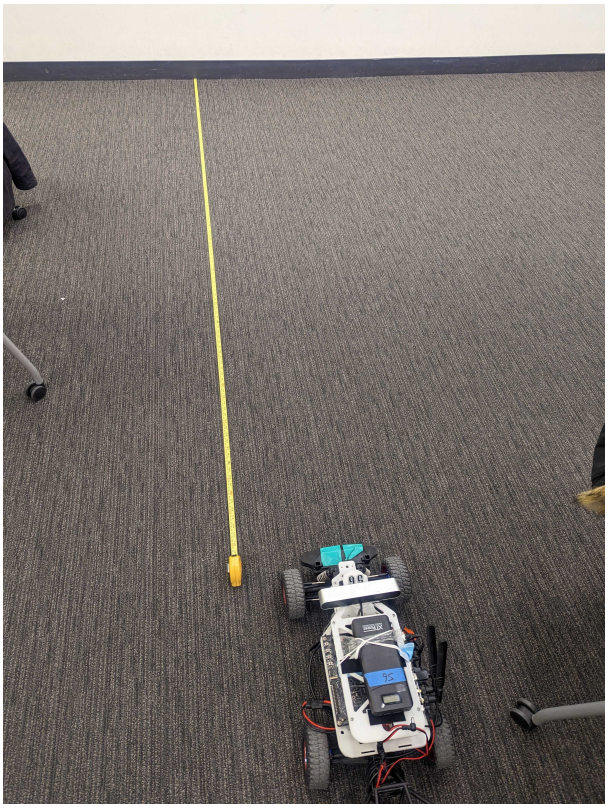


Figure 9: Setup of Safety controller test. Speeds of 1, 2, and 3 m/s were used

4 Conclusion

Authored by Vittal Thirumalai

In this design phase, we translated a simulated wall following algorithm to a physical, real-world robot. We learned how to effectively communicate with the physical robot using ROS. During the process, we encountered numerous obstacles as is expected when dealing with physical sensors, CPUs, and actuators. Through overcoming these obstacles, we learned a great deal about control and navigation, and we have now become decently comfortable with the racecar and ROS system.

There are some areas that we hope to work on to improve our wall following implementation. In particular, we noticed a small steady state error from the desired distance. However, the error was within a couple centimeters so we did not prioritize this compared to other tasks that we had to work on this week. To fix this, we would need to add the I term to our current PD control. This would require a decent amount of testing to be of value, but it could be beneficial in the long run.

Another area of potential improvement is the safety controller. Currently, it is only based on the distance directly in front of the car and the velocity of the car, but it could be enhanced by also taking into account the steering angle of the car. If the wheels are turned, the current look ahead collision detection algorithm is too conservative, so it is possible that the car may prematurely stop when turning if it gets too close to a wall. However, this very rarely occurred during testing if at all, and it is better to be on the safer side anyways.

Overall, we have built our wall following algorithm to be effective, safe, and portable. We look forward to combining navigation with computer vision in lab 4!

5 Lessons Learned

5.1 Alazar Lemma

I learned a lot this lab. For technical knowledge, I got the ability to learn more about ROS and git, two programming systems which I was not very familiar with With the help of my team, I became more familiar with these programs, and the syntax behind them. I, along with my team, experienced a lot of bugs and other issues all throughout the lab, so I got to practice persevering through those problems, however annoying they were. Employing PID control was really fun, as I got to hear out my teammates talk about their rationale, and bounce ideas off until we converged on a final solution.

On the communications and teamwork side of things, I got the opportunity to practice working with others, both in a team and among other teams. Withing my team, we had to figure out how to fairly distribute work, and coordinating times to meet that would be fair for everyone, during which we encountered headaches. The logistics of who is taking the robot home for the night, and who is bringing it where is very important given that not only your team, but other teams depend on the robots. Among other teams, I learned to coordinate dropoffs and pickups for the robot, as well as being flexible when other teams were struggling, and relying on other teams when we were struggling.

5.2 Benjamin

This lab taught me how to think critically about state machines and algorithms. From the very beginning, I struggled with creating my individual wall following algorithm in simulation. I'm not used to programming such difficult tasks without having skeleton code to follow or instructions on functions to implement. I spent many hours on the initial setup of my algorithm, simply deciding which functions I'd have to implement, how tasks would pass variables and data from one source to another, and how I might implement tests to avoid cascading errors. I'm proud of my simulation algorithm, which took advantage of the $y = mx + b$ equation I gathered from performing linear regression on the wall. Though it only implements p control, it amounted to tying in information from the velocity to judge error as the future distance from the setpoint. Though it didn't end up in our final real-world algorithm, deriving the formula enabled me to create the PID controller we did end up using in our final cumulative algorithm.

I also learned much about teamwork and communication. At the beginning of the lab, I felt I might have had a hard time contributing to the group since I have more experience with hardware than software. However, I learned to play to my strengths. My experience with control theory, MATLAB, testing, and statistics, allowed me to take lead on creating the PID control, parsing data, and debugging code. I leaned on others for help in ROS infrastructure, tuning the controllers, writing functions, and working through GitHub. Our team is very talented and members were willing to make a great effort to maintain flexibility in their schedules in order to finish this lab. Communicating about when we were going to be too busy to work on the robot, allowed us to plan ahead for specific work sessions.

5.3 Caden

This is the first time I have worked on a development team and working together taught me a lot. This was the first organization I have had on GitHub and I personally encountered many things in this lab that I have never seen before that are really exciting. Learning how to navigate and overcome adversity through clear communication was a crucial part of this lab and I am sure that I will have to execute great communication skills in future labs as well. When encountering a problem, we reasoned it out very effectively and one thing that I found worked well was taking turns when discussing how we thought we should solve a problem. It worked better to have one person propose one idea at a time instead of more than one person talking at a time which ultimately was a more efficient and effective method of problem solving.

On the more technical side of things, I learned an incredible amount of stuff about linux and middleware. Having never used either of the two in the past, it was a huge struggle for me to even get started but I pushed through it. I had many sleepless nights just trying to get our robot to work and I do have to say it paid off in the end. I learned a lot about how to dissect lidar data, and I also learned a lot about state machines. Having to think critically about these two things was essential for getting both the wall follower and the safety controller to work, and to work simultaneously. I had never worked

with lidar before, so this was new and exciting for me. In addition to this, Ben taught me another means of derivative feedback control for PID. Because I had only used previous error in my calculations, I had never thought about using a future projection like Ben had used and I found it really interesting.

In addition to this, learning to collaborate with other teams for robot use was a challenge that I think we all overcame quickly. We developed a system with other teams where we would meet up at designated locations and it worked pretty well. We all formed a large pod 6 messenger chat, nicknamed ourselves with our team number, and ultimately had an efficient communication system formulated entirely on our own. I was surprised at how well we were all able to collaborate, and I learned that people are so much more flexible and understanding than I could have ever imagined.

5.4 Henry Heiberger

As the first lab of the class where we actually were able to work with physical robots, this lab provided a wealth of problems and challenges that were very different from the previous two labs of the course. Through this process, I was able to learn a huge amount, making gains in all categories including my technical, communication, and collaboration skills.

Regarding the technical side of the lab, this lab was the first time where I was able to take on the task of implementing and tuning a controller completely from scratch on a physical device. This provided a huge opportunity for me to better cement my understanding of the control theory that I studied in previous classes and experience working with the design challenges that come when switching between simulation and the real world. Furthermore, the lab allowed me to explore the ways in which ROS operated on a real robot. This greatly increased my understanding of the middleware suite and will allow me to handle it more effectively in future labs. Paired with increasing my comfort with working in a Linux environment, this lab did a great job of improving my technical skills.

Regarding the communication side of the lab, the process of helping author a very detailed lab report and engaging briefing PowerPoint introduced me to the process of applying the communication skills I've gained through previous CI-H homework to a technical problem. This forced me to think about the problems I approached in new ways and put in the effort to figure out how to put them into words in an engaging and understandable way. This is a critical skill for a career in a technical industry, and this opening lab started the process of developing it. Paired with the briefing presentation which introduced me to a new methodology for presenting a PowerPoint in a way that is as engaging as possible, I made a huge amount of communication gains throughout the duration of this lab.

5.5 Vittal

This lab was a good learning experience for me. For the technical side of the lab, I gained a better sense of control theory, as I saw the PID control applied to a physical robot. I also got a better understanding of ROS and its use as middleware to interact between software and the robot. Furthermore, I realized the limitations of real-life sensors and the inherent measurement errors that they contain, which must be accounted for in code.

On the communication side of things, I learned how to create a technical report following IEEE guidelines. I also learned about delegating tasks within a team and splitting up parts in a way that makes the team collectively most efficient. Moreover, I gained experience with overleaf, creating visual diagrams, and incorporating them properly into a technical report.

Regarding collaboration, this lab was illuminating to me about learning to work well with a team in difficult circumstances. As I was out of town last weekend and have been feeling sick this week, I worked mainly virtually with the team. Given the situation, I communicated with the team primarily through

messenger and zoom. I contributed by discussing our initial approaches, helping with code especially with the corner overriding algorithm, and taking a larger role with making the design report and briefing slides. Overall, I have gained valuable lessons in technical areas, communication, and collaboration.