Lab 3 Report: Wall Follower and Safety Controller

Team 16

Jared Boyer
Daven Howard
Amy Ni
Niko Ramirez
Kayla Villa

1 Introduction - Niko

This paper presents our approach for foundational navigation capabilities within an autonomous racecar. Specifically, we aim to create autonomous driving algorithms that are robust, safe, and nonrestrictive to performance. In order to build the framework for these algorithms, this lab focuses on being able to have basic robotic control over variables such as steering angle, velocity, and path creation.

Our solution to this problem starts with a wall following controller that regulates the robot's velocity and desired distance from a wall on a given side of the racecar. The general overview of our solution to the robotic control problem starts with identifying surrounding walls and obstacles. Specifically, we utilize the LIDAR (Light Detection and Ranging) scanner equipped on our racecar to detect distances of obstacles in front of the car. To account for potential noise and uneven surfaces from the scans, we utilize least squares regression to create a regression line for the projected wall. With the code to handle LIDAR scans, the next step in our solution was to visualize these scans in rviz as we manually drive the car around. Once we ensured the car was working as expected, we began working on the actual wall following algorithm. Ultimately, we utilized a pure pursuit tracking algorithm to continually follow a created path based on the desired distance from the projected wall and the current racecar position.

The second component to our solution for a robust, safe, and nonrestrictive autonomous driving algorithm was implementing a safety controller. The goal of this safety controller was to prevent our racecar from crashing into both static and dynamic obstacles without hindering its performance. To achieve both of these goals, we implemented a system that checked a future projection of the racecar for possible collisions. Specifically, we estimate the future position of the racecar by multiplying its current velocity and a manually decided timestep. With this new position, we then create a collision zone around that single position to represent the entire car. If any scans enter this projected collision zone, we immediately stop the racecar by setting its velocity to 0. By only checking a small time into the future, we ensure the car isn't too conservative when approaching obstacles.

With our solution focused on wall following and a safety controller, we created a framework in this lab that we can iteratively build off for the rest of the semester. The work presented in this paper is the control and kinematics foundation for our autonomous racecar. Over the semester, we will add more sophisticated capabilities involving computer vision, perception, and planning algorithms. Ultimately, our racecar will be able to drive fully autonomously and compete at the end of the semester in a race.

2 Technical Approach - Amy, Jared

As previously mentioned, our primary objectives for this lab consisted of enabling the racecar to follow a wall and writing a safety controller to prevent collisions. Heading into lab 3, we were equipped with preliminary implementations of different wall following algorithms and their respective performances from the issued suite of test cases in lab 2. This proved to be invaluable information for our group's final implementation, as comparing initial performance allowed us to quickly hone in on the algorithm we wanted to pursue, which was Pure Pursuit. As for the safety controller, we aimed to create a simple, effective collision prevention controller. By focusing on collision avoidance in emergency situations, we were able to mitigate potential damage without compromising on the performance of our wall follower by being overly conservative. The following subsections serve to explain the technical details behind our wall follower and safety controller, as well as justify the design choices we made along the way.

2.1 Wall Follower

Designing a wall following algorithm is a specific case of a more general control task called *Trajectory Tracking*. In trajectory tracking, a robot has knowledge of its current pose, motion, and a desired trajectory, and its goal is to apply control effort in order to follow this trajectory as close as possible. The traditional feedback controllers we explored in class (PD, PID, etc.) are common solutions for simple trajectory tracking, but one of the highest performing algorithms is a controller called Pure Pursuit. In contrast to PID control, rather than tuning various gains depending on our system, Pure Pursuit exploits geometric relationships in car dynamics (hence classifying it a *geometric controller*), allowing for an intuitive, consistent, and reliable trajectory tracker.

Using our gradescope performance metrics as an initial evaluation, we saw that Pure Pursuit scored 97.7% in following accuracy compared to the average PID performance of around 90%. Using these results in combination with Prof. Carlone's nod to this algorithm's utility later on in the course, we decided on implementing Pure Pursuit as our wall following algorithm.

Before tackling the technical intuition behind Pure Pursuit, it's important to first investigate how the robot actually generates the trajectory to be followed, which is ideally a line offset from the wall.

The robot first receives LIDAR data, detailing where it detects obstructions with respect to the robot coordinate system (Illustrated in Figure 1). Since we are only interested in following a wall on one side of the robot at a time, we filter out points that aren't within a region of interest. That is, following a wall to the left of the robot shouldn't need any information on obstructions located to the right, and vice versa. We decided to choose a 90° quadrant as our region of interest, analyzing the wall directly adjacent to the robot, as well as accounting for potential corners that lie ahead. Additionally, we restricted the robot to only consider points within a maximum distance of 2.5m of the robot, preventing far away points from skewing our estimate of the wall.

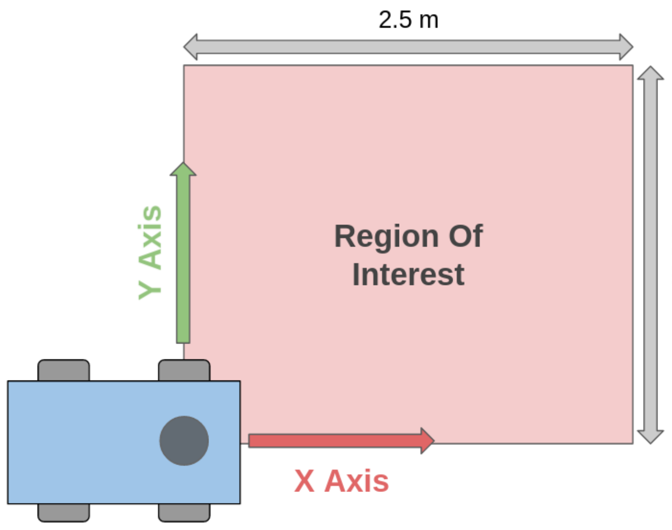

Figure 1. Racecar LIDAR Region of Interest

After receiving LIDAR data and restricting the set of points to analyze, our next job was to fit a valid trajectory to our observations. By assuming the wall is roughly straight, we decided to perform a simple linear regression (using numpy's *polyfit*) to express our observation of the wall as a line. This also had the useful side effect of filtering out measurement noise, as fitting a line to a set of points has the intuition of averaging these points out. Since the wall estimate was expressed in slope-intercept line form, where the intercept represents the robot's instantaneous distance from the wall, converting this to a desired trajectory was as simple as adding (or subtracting) the desired offset from the wall to the equation's y-intercept:

$$y = m * x + b \qquad\qquad y = m * x + b + y_{off}$$

Equations 1 & 2: LIDAR Fitted Line and Its Corresponding Offset

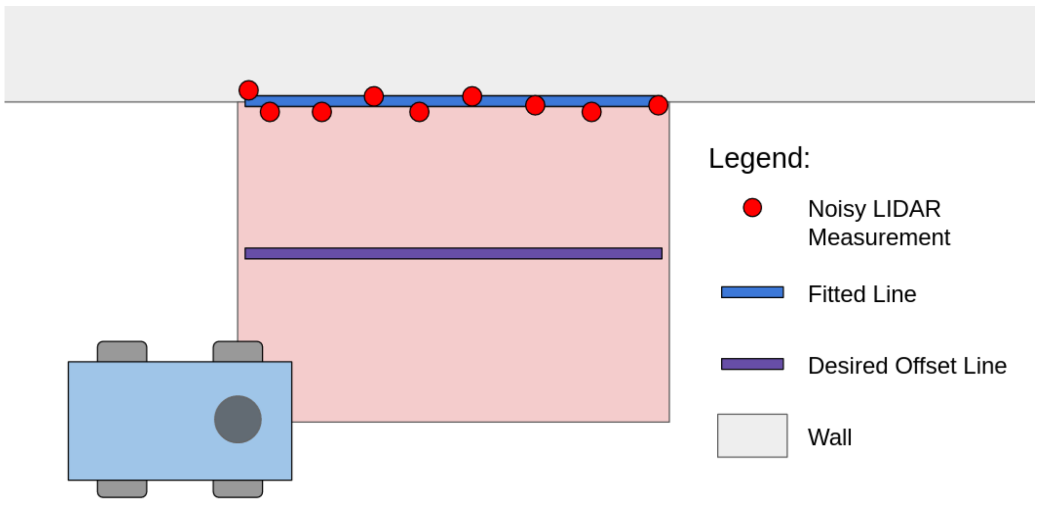

Figure 2: Racecar LIDAR Line Fitting

Now that the robot has a desired trajectory to be followed, Pure Pursuit can be used to derive the best driving commands for tracking. At each point in time, the controller starts off by selecting a point on the desired trajectory such that this point is one *lookahead distance* away from the robot. This calculation boils down to finding the intersection between a line and a circle, because the robot's locus of points that are one lookahead distance away inscribe a circle, and the desired trajectory is simply a line. For the sake of future iterations, which may involve finding goal points on a potentially non-linear trajectory, the algorithm we implemented didn't assume our path was necessarily a line. By utilizing an intersection algorithm found in this online forum, our implementation is also capable of calculating goal points in the more general case.

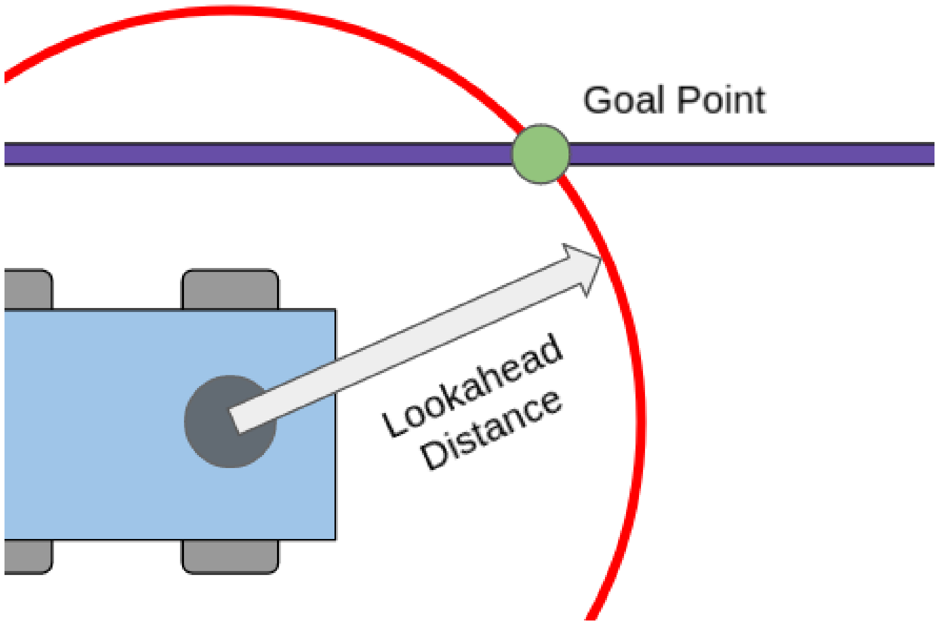

Figure 3. Calculating Goal Point as a Function of Lookahead Distance

After finding the location of the goal point, an arc connecting the robot and this point is constructed. The radius of curvature of this arc can be calculated based on simple geometric relationships introduced in class, and given this radius of curvature, we can also find what steering angle should be applied. These equations are derived from the standard bicycle model, which can fully characterize the motion of the racecar given the car length, steering angle, and velocity.

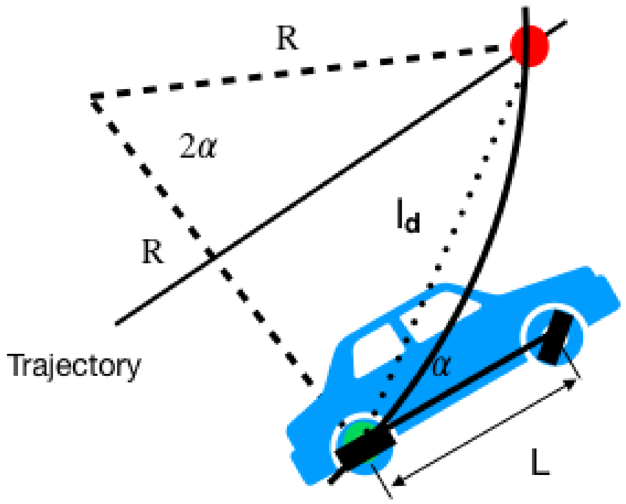

Figure 4. Pure Pursuit Geometric Relationships

$$R = \frac{l_d}{2 * sin\alpha} \qquad \delta = \arctan\left(\frac{L}{R}\right)$$

Equations 3 & 4: Radius of Curvature and Steering Angle Calculations

By repeating these steps at each point in time, Pure Pursuit results in robotic motion that is both smooth and accurate to the desired trajectory. Another point to be noted is that the lookahead distance of Pure Pursuit is the primary parameter that can be tuned. Smaller distances correspond to fast, aggressive behavior that may suffer from stability issues, whereas larger distances correspond to smooth, stable, but slower performance.

Ultimately, our Pure Pursuit implementation showed very promising performance in our tests and evaluations. That being said, there are still some areas for improvement. Tuning the lookahead distance can result in a better performing controller, but most notably, more work needs to be done to optimize performance in the controller's edge cases. For example, if the robot doesn't detect any LIDAR points in the region of interest, our implementation currently lacks a robust system for re-orienting the robot.

2.2 Safety Controller

The main purpose of the safety controller is to prevent any potential collision the car might encounter, eliminating the possibility of damaging the racecar and its onboard equipment as well as preventing collisions with humans. In addition, the controller should not hinder the car's regular navigation tasks and only trigger under emergency situations. With these two main goals in mind, our team developed a projection system based collision avoidance controller.

As the car is driving at a certain velocity, our safety controller will estimate where the car will be after a short time interval into the future and use this projection to predict incoming collisions. When the car's onboard LIDAR detects the surroundings, the safety controller compares the LIDAR data with the projected location of the car to check whether any data point collides with the projection. If a LIDAR point intersects with the projection, it can be reasoned that the car will crash into the obstacle within the short time interval that we used to define the projection.

To find the exact coordinates of the projection, the controller takes the current velocity of the car and multiplies it with our manually defined time interval. In this case, we defined the time interval to be 0.5 seconds, found through manual testing of various values. These calculations are done with respect to the car's coordinate frame, and since the car is driving forward along the x-axis, the (x,y) coordinate of the car's position in 0.5 seconds will simply be:

*coordinate = [velocity * 0.5, 0]*

Given this point coordinate, we then modeled the projected car as a circle centered at the projection coordinate with a radius of 0.15 meters, which is roughly half the length of the car. We defined this circular region as the “collision zone” that the safety controller uses in combination with the LIDAR data points to check for potential collisions. If any LIDAR points are detected within the collision zone, the safety controller recognizes that there is an incoming crash and stops the car immediately by setting the velocity of the car to 0 m/s, preventing said crash. Figure 5 provides a visual representation of the projection approach.

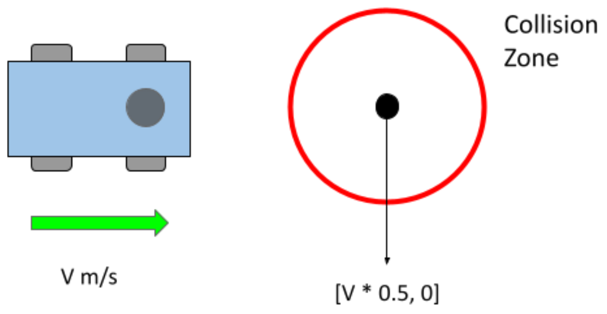

Figure 5. Demonstration of safety controller concept

However, as we were testing the safety controller by driving the car towards obstacles at various speeds, we found that the controller was not well suited to handle high velocity operation. Since the x-coordinate of the projection point is proportional to the velocity of the car, the distance between the physical car and the projection increases as velocity increases. As a result, when the car is operating at a high speed there is a chance for an obstacle’s LIDAR points to lie in between the car and the collision zone, causing the safety controller to miss the obstacle and allow the car to continue at its original speed. Figure 6 demonstrates a scenario where the safety controller will fail to stop a collision.

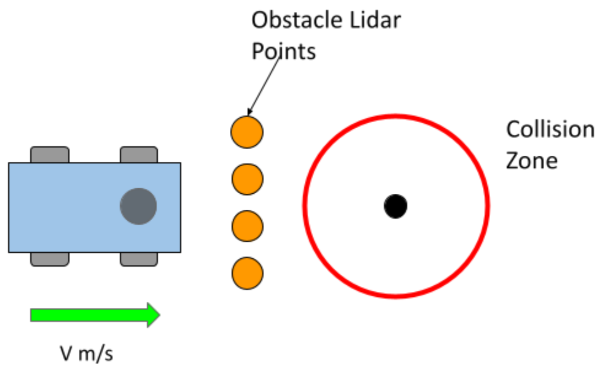

Figure 6. Safety controller fails to detect obstacle lidar points

To address the aforementioned issue, we modified our circular modeling of the projected car to include the space inbetween the projection point and the physical car itself. Instead of a circular representation, we defined a rectangular region that extends to the front of the projected car as the new collision zone. Figure 7 demonstrates the updated version of the projection concept. Similar to our previous model, the updated safety controller will still check the LIDAR data to see if any data point lies within the collision zone and react accordingly.

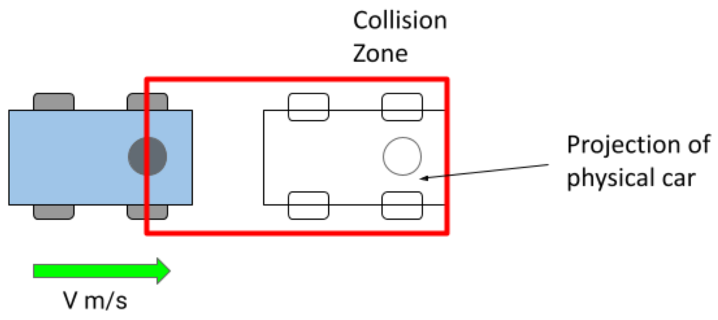

Figure 7. Demonstration of the rectangular collision zone model

Although our safety controller can now handle higher velocities with the new rectangular collision zone model, it is still vulnerable under scenarios where the car is not driving straight forward. Because the collision zone is calculated from a linear estimation of the car's future coordinate, it is not robust enough to accurately predict crashes when the car is turning at a significant steering angle. In addition, the controller simply stops the car until the obstacle disappears when it detects a possible collision. In the future, we would like to improve the safety controller, allowing it to estimate the car's path more precisely and possibly backing out of a collision to bypass the obstacle.

3 Experimental Evaluation - Daven

In order to test our racecar, we decided on a few tests and metrics in which to evaluate the racecar's performance and determine how well it met the goals of the lab. For instance, to evaluate our wall follower, we defined several scenarios we wanted our racecar to successfully navigate. We determined whether or not the car was successful by its ability to follow the wall at the desired distance. These were qualitative tests and judged performance on whether or not the car completed the task from an objective perspective. We also performed a quantitative test, measuring error which we defined as the difference between the desired distance and the car's true distance from the wall as it followed a straight wall. In these tests, we analyzed the racecar's steady state behavior as well as the time taken to reach the desired distance from various initial conditions. That being said, most of our testing focused on qualitative scenarios, and more work needs to be done to better characterize the racecar's quantitative performance. Our tests for the safety controller were limited to two qualitative tests to determine the racecar’s ability to stop safely. In the future, we would like to expand on this test suite and develop more tests for our car’s safety controller.

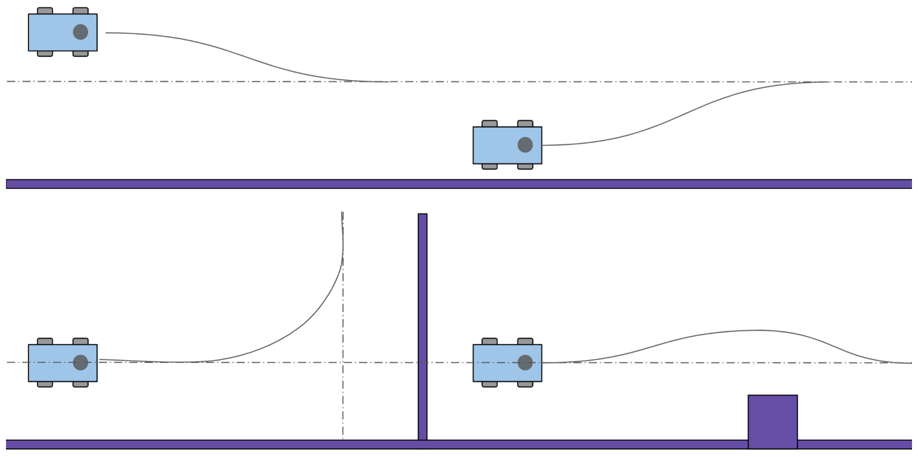
Figure 8: Four scenarios for wall follower test

Our qualitative testing for the wall follower was composed of 4 different scenarios where we determined if the racecar successfully followed the wall at a predetermined desired distance. The first test was to check the ability of the car to reach the target distance when it started from a distance greater than the target distance. The second test was similar and checked the car when it started at a distance less than the target value. The combination of these two tests evaluated the ability of the racecar to reach and stay at the target distance when starting from an arbitrary distance from the wall. The next test analyzed how well the racecar navigated around a corner while remaining at the target distance both before and after the turn. Our last test determined the car’s ability to drive around an obstacle jutting out from the wall. This test was important to ensure the racecar followed the wall despite any contours and evaluated its ability to maneuver around obstacles that may be blocking its path. Since the wall follower can be set to follow either the left or right wall, although our system is symmetric, we ran each of the four tests for each side.

The quantitative testing for error consisted of running our racecar with the wall follower along a long straight wall. We chose an arbitrary speed and desired distance in our tests since we wanted a more general idea of the racecar’s behavior. In order to gather data, we used a rosbag to record two topics, one with the desired distance from the wall (constant), and the other with the actual distance from the wall based on the LIDAR data. Gathering data during the run in this way allowed us to easily visualize the data in a graph as seen in *Figure 9.*

The two tests we performed for the safety controller ensured the racecar’s ability to stop for both static and moving objects in its path. The first test had the racecar drive towards an obstacle that was already directly in its path, checking that it would stop appropriately before crashing. The second test was similar but instead of the obstacle already in the car's path, we dropped the obstacle in the car's path a small distance in front while it was moving. Both of these tests were evaluating if the racecar would stop before hitting the objects at a small distance away while not being too conservative which could affect its ability to perform other tasks.

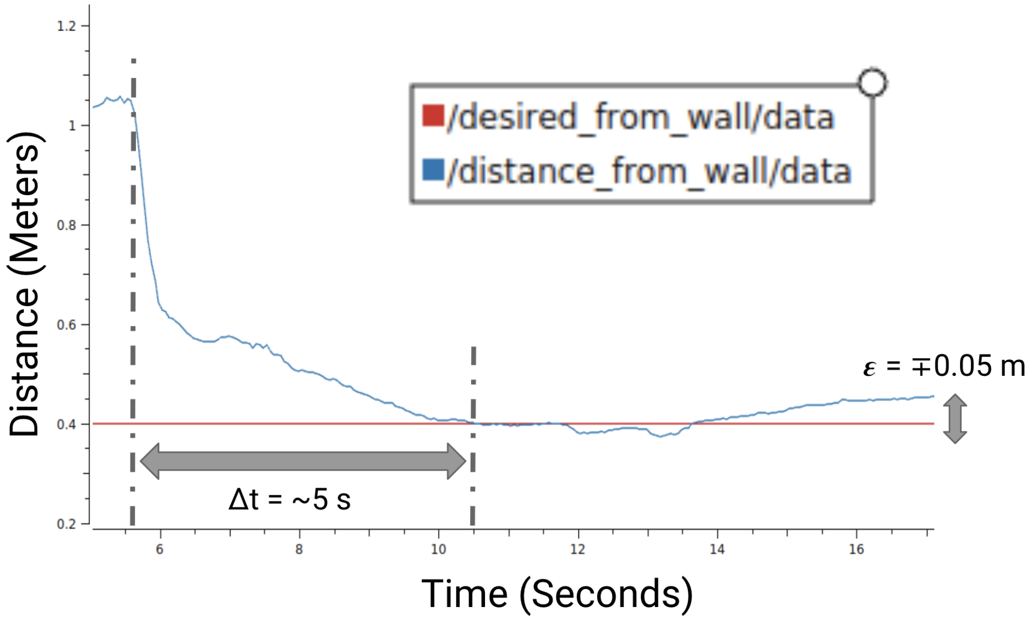

Figure 9: Graph comparing set distance and actual distance from the wall

Collecting the results from our tests, we found that the racecar performed quite well. From an objective standpoint, the racecar passed all four of the qualitative tests for the wall follower. In each test, it was successful at following the wall at the set desired distance and navigating each of the scenarios. The data from the quantitative test of the wall follower can be seen in *Figure 9* which compares the set desired distance and actual distance from the wall over time as it traveled along a straight wall. From this graph, we can see that once the racecar reached steady state, the error in the distance from the wall was about +/- 0.05 meters. We can also extract from the graph the time taken to reach the set desired distance as about 5 seconds. This graph only shows data from one run of the test but we found similar results in the other tests. In each of the two tests for the safety controller, the racecar passed, stopping safely and preventing itself from crashing.

Analyzing the data we can confidently say our racecar met the primary goals we set out to accomplish. The racecar successfully followed a wall in various tests and prevented collisions with obstacles. From the quantitative data, we see that the car maintains a small error of +/-

0.05 meters when following a wall at a set distance away which is within an appropriate tolerance given our hardware and goals. The time to reach the desired distance was sufficient although we do believe this can be improved and the amount of time lessened. These results showed the car was not turning aggressively enough, likely due to the lookahead distance in our pure pursuit algorithm being too great. This is something that can be tuned and hopefully improved as the racecar is developed further.

4 Conclusion - Kayla

In this lab, we were able to successfully build navigation capabilities onto our car that allowed it to follow a wall on a specified side of the car at a certain distance, as well as implement a safety controller that would stop the car in the event of an impending collision. For our wall following code, we decided to use the Pure Pursuit algorithm because of its ubiquitous use in the robotics industry as well as advice from Professor Carlone that using Pure Pursuit for navigation would be helpful in future labs. For our safety controller, we decided to use a linear approximation of the car's path at a time point in the future to determine if it was going to be involved in a collision at that point in time, and stop if so. We tested both our wall follower and safety controller code by placing the car in various scenarios of differing properties, such as the starting distance and angle from the wall, shape of wall, velocity, and presence of obstacles. Based on our car's performance in our test suite, we are confident in our car's navigational abilities to follow a potentially non-uniform wall as well as its ability to stop before an impending collision.

As a team we identified some areas of improvement that we can implement as we continue to refine our car for the final challenge. We can envision a scenario where it would be beneficial for the car to be able to try navigating around an obstacle again if it wasn't able to do so successfully the first time, so that it can continue on its path. Therefore, we think it could be advantageous to modify our safety controller so that the car reverses out of a collision pathway and tries to navigate around the collision again, instead of stopping altogether.

We also think that the way that our code predicts the car's location at a future time step could be improved to have higher accuracy. We currently use the car's velocity to calculate a linear approximation of the car's location at a time step into the future. However, we think that accuracy could be improved by using a higher order approximation for the path of the car that takes into account the steering angle, steering speed, and acceleration of the car. This could potentially allow us to calculate a more accurate model of the car's future path and therefore have a more precise idea of the car's future location. This would lead to more accurate collision detection and path following.

By implementing navigation and safety capabilities into the car, our team has taken a strong first step towards building a successful autonomous racecar. As we continue to refine our car for the final challenge, these ideas highlighted above will be improvements that we try to implement into our car to make it as successful as possible.

5 Lessons Learned

5.1 Kayla

Communication: I learned that it was very beneficial when problem solving to discuss the high level ideas for solving a problem while we were together in lab or in our meetings outside of class, and then work on implementing the details of that solution while we were apart. This way, we optimized the time that we were able to spend together to ideate towards finding solutions to the various problems in the lab.

Technical: I learned the importance of building capabilities into the car in a modular fashion, by first starting with building basic controls into the car such as having it follow a straight wall, and then progress from there by introducing more complex capabilities such as navigating around a turn. By starting with the simpler tasks first and building up from there, we were able to build capabilities incrementally which helped with debugging our code.

5.2 Niko

Communication: I learned the true value of organization and division of work to be efficient as a team. Specifically, I found splitting up work during the labs, then meeting at a specified time every week (Sunday for us) allows us to easily keep up with the tasks. This also enables us to do the work at our own pace and time, then meet together to combine, debug, or design new ideas. In addition to Sunday meetings, last minute remote meetings were super useful to finalize any details.

Technical: I learned how powerful iteration is for building a very complex system such as the racecar. In this lab, we saw success by starting on simple implementations and continually updating on it. This enables us to be both efficient in our short time and effective in achieving our goals. The class also takes a similar structure as mentioned in lecture, work on getting the racecar to work on hardware, then iteratively add more complex systems such as computer vision.

5.3 Jared

Communication: In comparison to previous group projects I’ve encountered, I put more effort than usual into having deliberate and respectful group discussions. Specifically, I tried to focus on listening and being open to other’s thoughts with the goal in mind to make our team as collaborative and inviting as possible. I think our team dynamics were a success and we excelled in communicating effectively and respectfully, and as a result I plan on listening, being respectful, and valuing my group member's opinions as much as possible moving forward.

Technical: I learned that for building a complex autonomous system with many working parts, it's often beneficial to start with simple solutions. Rather than attempt to create a system that's overcomplicated from the start, it's often useful to create a working implementation of the most basic version of a technical solution, and then from there work on iterating and improving its capabilities. I think this is one of the things that went very well for our team - by selecting achievable and simple goals (with our safety controller, for example), we were able to progress quickly and efficiently.

5.4 Amy

Communication: One method that was really helpful for the team was to discuss the high level idea together before splitting up to come up with our own individual implementation drafts. Agreeing on a general approach helped us to focus on a common direction when we came together to discuss future steps. In addition, setting aside a significant block of time to meet up over the weekend in order to talk over the code and debug together made the collaboration process much easier. Overall, I really enjoyed our team dynamics and work process so far, and I hope we can keep up the synergy for all future labs.

Technical: In past group assignments, it is common for my team to aim for a complex overall solution and split up the deliverables accordingly. For lab 3, I really appreciated how my team decided to start with a simple and straightforward solution. As we focused on implementing the initial draft of the code, I realized how it gave us a better insight into the problem we are solving and made it easier to think about possible improvements for a more extensive and robust version of the code.

5.5 Daven

Communication: I found that choosing set times early on to come together and work for a few hours was beneficial to both our team dynamic and work. It allowed us to discuss ideas with one another from the beginning to begin our work from a shared and developed perspective. This aided in us all being on the same page as we completed the different parts of the lab. I learned how valuable it is to meet early and set a good foundation for how things will work for the rest of the project.

Technical: Many times I find in group projects we all want to create the best product from the beginning and get carried away with complexity. This time I learned how beneficial it is to start small with a basic idea before moving into more complicated and time-consuming ideas. While discussing our safety controller we all had great ideas but realized they may be outside the scope of necessity. This enabled us to save time by implementing a simple controller which turned out to work very well.