Lab 3 Report: Wall Following on Racecar

Team 19

Michael Wong
Alan Yu
Joshua Sohn
Owen Matteson
Ragulan Sivakumar
Shenzhe Yao

6.4200: RSS

March 11, 2023

1 Introduction (MW; edited by OM)

Wall following is one of the fundamental baselines for an autonomous racecar. Our purpose in this lab was to implement a wall following algorithm for an RSS racecar by utilizing the data published by the mounted LiDAR scanner (Figure 1). We planned for this algorithm to be capable of handling noise and unpredictability in the LiDAR representation of the environment. Furthermore, we saw it valuable to add safety capabilities to the system in order to assist in avoiding collisions with dynamic and static obstacles. We were able to utilize the group members' various algorithms from Lab 2, wall following in a simulated environment, in order to accomplish this goal.

As a group, each of us had unique pipelines and strategies to tackle the wall-following in simulation problem. Starting from a baseline pure PID controller, we pieced together unique ideas while also acknowledging and accounting for the weaknesses of each.

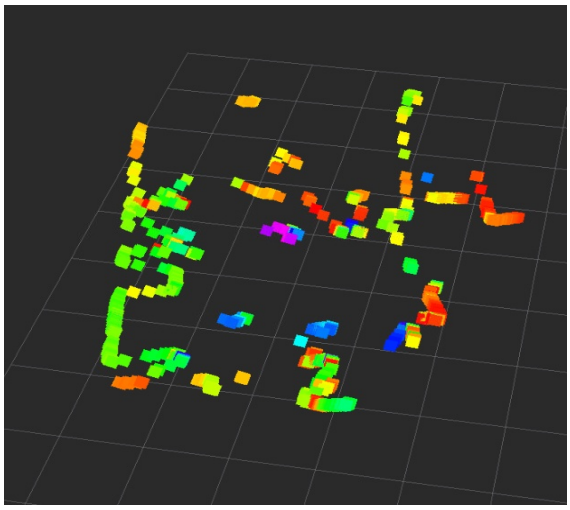
To improve our car, we discovered three main problems to address.

Our initial challenge was dealing with the car oscillating left and right. We were able to follow a straight wall, but starting angled away caused a series of over-corrections and infinite oscillations.

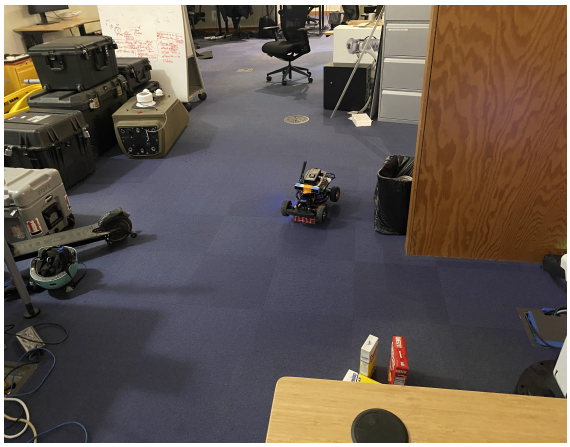
Another obstacle we faced was detecting and navigating corners effectively. With the pure PID controller, we could execute sharper turns by increasing the scan range. However, this was insufficient and we had to turn to a separate corner detection algorithm.

Because the racecar's actions were at first unpredictable, it was also necessary to implement a safety controller. The safety controller works in the background, continuously monitoring the car's movement and taking over autonomous control if necessary. If the car approaches an obstacle within a certain distance, the safety controller's primary objective is to bring the car to a stop and prevent any damage.

For each of these issues, we designed an individualized test case to quantify how much our changes improved the algorithm. We also developed an overall test suite to evaluate the improvement between the PID controller and our final product, *M.A.J.O.R.S.*



(a) LiDAR Scan



(b) Car in Real World

Figure 1: Visualization of the RVIZ LiDAR scan (a) associated with the car's position in the real world (b). The lasers identify objects surrounding the car and exhibit some noise.

2 Technical Approach (2500 words)

2.1 Problem Formulation (RS)

In this lab, our goal was to create two robust algorithms for our racecar: a wall-follower and a safety controller. For the wall-follower, we wanted our racecar to follow a wall on the right or left-hand side at a given desired distance, δ, away with small steady-state error from said desired distance. We also wanted our racecar to turn appropriately at corners. For the safety controller, we wanted our racecar to only stop when it does not have enough time to avoid an obstacle, for this would prevent damage to our racecar and the obstacle impeding our path. With these goals in mind, we took the following initial approaches in coming up with our wall-follower and safety controller.

2.2 Initial Approaches (RS)

2.2.1 PID Control

Before progressing, let us describe a technique employed within our wall-follower: PID control. PID stands for proportional, integral, and derivative. These terms are in reference to the measured error between our actual system and what is desired. For example, we multiply our error by constant K_p for proportional control, the derivative of our error by constant K_d for derivative control, and the integral of our error by constant K_i for integral control. We sum those three values to get the command we send to our system actuator. If $K_p, K_d,$ and K_i are chosen well, this feedback loop will converge towards our desired behavior.

2.2.2 Wall-Follower

Each member of our group came up with an implementation for a wall-follower in simulation. Thus, we first tested each implementation on the racecar. The two most promising implementations were Design A 2, a P-controller whose K_p value varied inversely with racecar velocity, and Design B 3, a traditional PID-controller that was better at navigating around walls.

2.2.3 Design A: P-controller with velocity-dependent K_p

With this P-controller, our procedure was as follows:

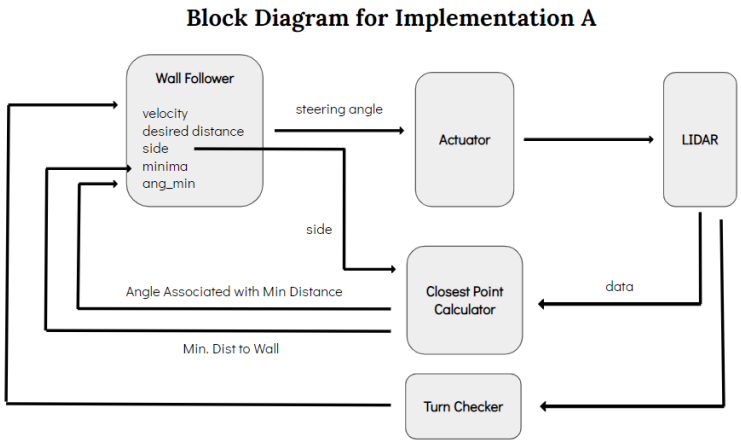


Figure 2: Initial implementation using velocity-dependent gains.

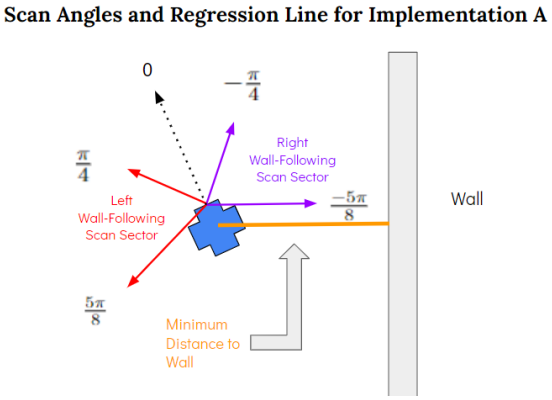


Figure 3: Illustration of scan angle partitioning.

1. Scan the LiDAR data on the side we are wall-following for the minimum distance to the wall, *minima*, and compute the angle, *ang_min*, of the vector *minima* with respect to the racecar.

2. Compute
$$ang\_cmd = \sin^{-1}(\frac{\text{minima} - \text{desired distance}}{\text{velocity}})$$
This is the angle we should go at to eradicate our error in 1 second if we were already driving parallel to the wall.

3. Compute ref_angle.
   - Wall-Following on Right: $ref\_angle = ang\_min + \frac{\pi}{2}$
   - Wall-Following on Left: $ref\_angle = ang\_min - \frac{\pi}{2}$

   This calculates how far our current driving angle is from parallel to the wall. This exploits the fact that we want the minimum distance to come at angle $-\frac{\pi}{2}$ for wall-following on the right and $\frac{\pi}{2}$ for wall-following on the left, for these would mean that we are driving parallel to the wall.

4. Compute
$$\omega = \frac{ang\_cmd + ref\_angle}{2 \cdot velocity}$$
where ω is the steering angle command we send to the actuator. The angle is divided here by $2 \cdot velocity$ so as to prevent the system from overshooting our desired distance and getting into oscillatory behavior. Effectively, this makes our K_p value inversely proportional to velocity.

5. However, if our turn checker module indicates that there's a wall ahead of us, override *steering_angle* and instead set it to the maximum angle away from the side we're following.

2.2.4 Design B: PID-controller

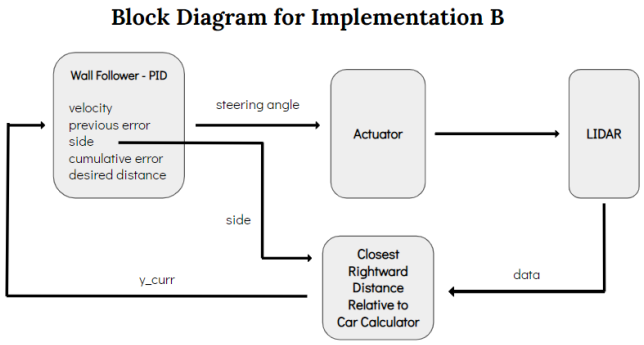


Figure 4: Pipeline for the velocity-independent controller.

With the traditional PID-controller, our procedure was as follows:

1. Scan the LiDAR data on the side we are wall-following. Compute a regression line for the wall.

2. Find the closest point by taking the absolute horizontal distance from this regression line relative to the car after converting from polar coordinates. Denote this horizontal distance as y_curr.

3. Compute error as the following:
$$\text{error} = \text{desired distance} - y\_curr$$

4. Maintain an integral and discrete derivative term for error. Multiply the integral of the error by K_i, derivative of the error by K_d, and normal error by K_p, and then sum them to get what the new steering angle should be. The K_p for this design was fixed at 0.80, and the other coefficients are discussed later.

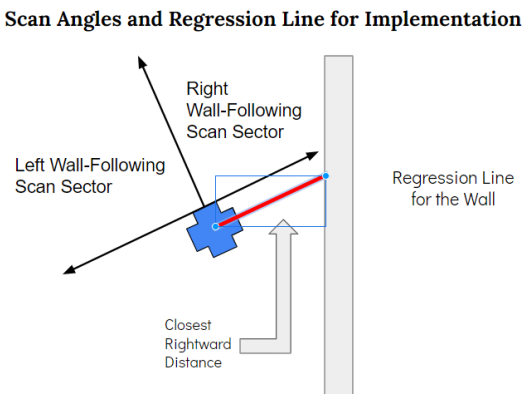


Figure 5: We partition the scanning regions into the left and right sides and regress to estimate the nearest wall.

2.2.5 Safety Controller

For the safety controller, our initial idea was that if an obstacle was within a certain distance in front of our racecar, then we should stop the car by turning its velocity to 0. However, we also planned to optimize this stopping distance so as to make sure we were only stopping when we absolutely needed to.

2.3 Main Problems

2.3.1 Oscillations (JS)

While all of our individual code seemed to work well in the simulation tests, we immediately ran into problems with following a straight line. All of our controllers were prone to steady-state error, meaning that the car would oscillate from side to side while trying to follow a straight wall rather than travelling in a straight line.

Increasing I and D Control Our initial reaction was to increase K_i and/or K_d, which would increase the effect of integral and/or derivative control respectively. Our intuition was that proportional control often leads to steady-state error, and integral control would improve steady-state error while derivative control would improve the stability of the response. However, upon experimentation, we found that increasing K_i by a non-negligible amount hindered the car's ability to to respond to turns in time. For K_d, we empirically determined an initial value of 0.10 that continued to be optimal for the rest of the lab.

Combining Design A and Design B Our solution was to combine Design A and Design B. We empirically determined that there was an inherent trade-off between Design A and Design B. Using Design B made turns more efficient but increased the steady-state error while following a straight wall. Using Design A made it easier to follow a straight wall but decreased the car's reaction for turns. Therefore, we decided that our final error signal should be a convex combination of the error signal from Design A, ω_A and the error signal from Design B, ω_B. Through experimentation, we find the optimal relative weight α^* between these two error signals, giving our overall error signal

$$\omega^* = \alpha^* \omega_A + (1 - \alpha^*)\omega_B, \tag{1}$$

where α^* is defined as the optimal *Ragulan Coefficient*, which the group named after one of its members, Ragulan Sivakumar, who was responsible for the creation of design A.

Ridge-Regularized Least Squares (AY) Qualitatively, we also noticed that the regression line was unstable, even when only following a straight wall. In order to add more smoothness to the estimate,

we introduce an ℓ^2 regularizer in the least squares objective. Hence, given scans $\{(x_i, y_i)\}_{i=1}^N$, we stack $\mathbf{y} = (y_1, \ldots, y_N)^T$ and $\mathbf{X} = \begin{bmatrix} x_1 & 1 \\ \ldots & \ldots \\ x_N & 1 \end{bmatrix}$, the new regression optimizes the objective

$$\min_{\beta \in \mathbb{R}^2} ||\mathbf{y} - \mathbf{X}\beta||_2^2 + \lambda ||\beta||_2^2, \tag{2}$$

where λ is the regularization parameter. This gives the closed form solution

$$\beta = (\mathbf{X}^T\mathbf{X} + \lambda I)^{-1}\mathbf{X}^T\mathbf{y}. \tag{3}$$

Furthermore, we filtered out outliers by only considering values within $\mu \pm 1.5\sigma$ for both the x and y distances from the wall before calculating the ridge regression, where σ is the standard deviation and μ is the mean. This allowed the regression to ignore points that were far from the racecar in order to not skew the fit.

2.3.2 Approaching Corners (AY)

There are two types of turns we want our wall-follower to be able to make: inside and outside ones. Inside turns are where the car turns on the inner 90° of a corner, whereas outside turns are where the car turns on the outer 270° of a corner. Our racecar will run across both of these in real life, so it is essential that it handles these turns properly.

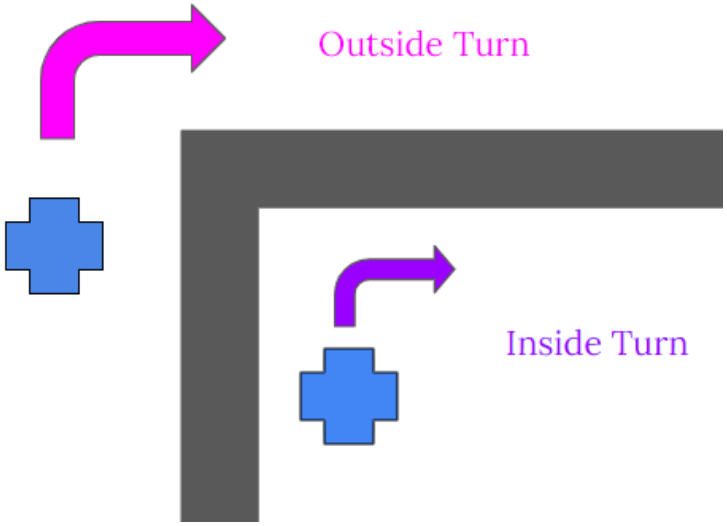


Figure 6: An illustration of inside and outside turns. Inside turns occur on the interior of the intersection of two walls, while outside turns navigate around the exterior.

Design B followed straight walls with strong oscillation from the desired distance but was better at making outside turns. Meanwhile, Design A handled straight walls and inner turns well (due to the built-in corner detector override), but it would get confused on outside turns. It would also struggle when the turn was tighter and had a much higher deviation from the desired distance than the PID-controller. Thus, our goal was to figure out how to combine these two implementations into a more-robust wall-follower.
Up to now, Design B's navigation around corners relies on purely the regression line and PID controller response. When approaching an inside corner (Figure 6), increasing the scan range $\Delta\theta$ results in a sharper turn. However, we find that a large $\Delta\theta$ is at odds with stability, since reading too far ahead can pick up points not on the wall. Hence, we restrict $\Delta\theta$ to a smaller range of $\frac{\pi}{3}$ and instead draw upon Design B's separate step for corner detection.

Inside Turns Our first attempt at resolving this issue was to estimate a good turning distance as a function of speed using a linear model. To do this, we fixed a desired distance δ and empirically determined n pairs of tuples (v, d), where v is the car velocity and d is the threshold for turning. We regress $d \sim v$, to get the estimate $\hat{d}(v) = \hat{\beta}v + \hat{\alpha}$.

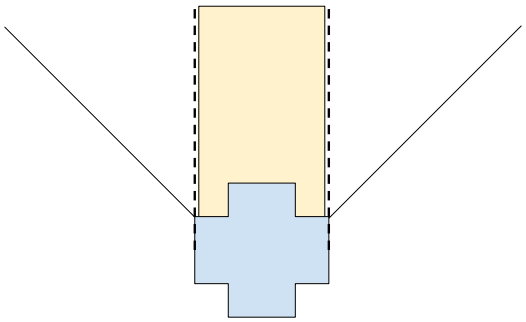


Figure 7: The viewing region of the car as it approaches a wall for turning. We filter points outside the width of the car.

The car detects if d is attained by scanning the rectangular space in front of it (Figure 7) and applies the maximum steering angle ϕ when reached. However, as we varied δ at test-time, we noticed that our predictions failed to generalize.
In our next attempt, we realized that in a perfect environment with zero latency, the velocity of the car should not affect its turning radius r. In fact, the variation in d should be a consequence of δ and ϕ, as shown in Figure 8, which we failed to account for in our first attempt. There is also some dependence on the vehicle dimensions. Specifically, let L be the length of the vehicle. Then, as illustrated in the figure, we have

$$r = \frac{L}{\sin\phi},$$

so

$$d = r + \delta = \frac{L}{\sin\phi} + \delta.$$

This result seemed to generalize better across different values of δ but was turning too late for high values of v. We concluded that this must be due to latency in processing the LiDAR scan data, so we include a linear term γv to account for such noise. This gives the final inner corner turning threshold of

$$d(v, \delta) = \frac{L}{\sin\phi} + \delta + \gamma v, \tag{4}$$

where we determine γ empirically. Unsurpisingly, the optimal value of γ is closely related to $\hat{\beta}$, the slope of our original regressed predictor function, since in both cases d is assumed to be linear in v.

Outside Turns Because the algorithm for inside turns depends on detecting a front wall, it does not extend naturally to outside turns. In particular, we noticed that if there is a narrow hallway with two walls close together , the robot will fail to recognize the hallway and instead turn along the far wall (Figure 9). Hence, with the above implementations, the robot will not reliably navigate outside turns. To fix this, we

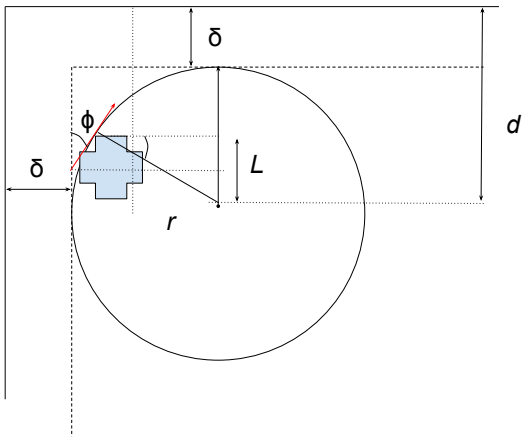


Figure 8: Figure showing how to compute the desired turning threshold d. The red vector indicates the direction of the outer front wheel, giving the relationship $L = r \sin \phi$.

Figure 9: An outside turn scenario where the close proximity of the far wall provides a false positive for the inside turn logic.

include an outside turn detector.

We observe that the main characteristic of an outside turn is the presence of empty points on the desired turning side. To recognize this case, we add another layer to the inside turn logic (see Algorithm 1). If the maximum distance on the desired side exceeds the threshold η from a wall, indicating an absence of a wall, then we have identified an outside turn and ignore the inside turn command, running PID uninterrupted.

d $\leftarrow \frac{L}{\sin\phi} + \delta + \gamma v$
x $\leftarrow \min(front\_ranges)$
if $x \leq d$ **then**
  if $\max(side\_ranges) > \eta$ **then**
    return *False*
  else
    return *True*
  end
else
  return *False*
end

Algorithm 1: Corner Detection Algorithm

2.4 Safety Control (JS)

Initial Approach Our initial approach was to scan angles that roughly corresponded to the front of the car (Figure 10.1). We arbitrarily chose the threshold values $[-\frac{\pi}{6}, \frac{\pi}{6}]$. We decided that the car should stop as soon as the minimum range within these scanning angles is less than a certain stopping threshold. For now, we set this stopping threshold to be a function of desired distance and velocity.

Scanning 1.0 We realized that directly slicing the LiDAR scan data was incorrect. Because the LiDAR scans are discretized by angle, it will sometimes detect objects to the side that are not directly in front of the car. Therefore, our first iteration converted the polar scan data into a Cartesian coordinates system of (x, y) points (Figure 10.2). By constraining the y values to be the within the width of the car, we are only looking at objects that are actually in the car's path.

Scanning 2.0 However, we ran into another issue during turns where the car would stop because it sensed a wall in front of it while it was turning that would not actually obstruct the cars path. We solved this issue by implementing an inverted triangle, where the scanning region decreases linearly as a function of the distance from the front of the car (Figure 10.3).

Stopping Threshold Finally, the stopping threshold was incorrect. First of all, desired distance should not affect the stopping threshold at all. Just because the car wants to be closer or farther from the wall has no impact on whether the car will crash or not. However, we were correct that velocity should determine the stopping threshold. In a perfect world, the stopping threshold would be a very small constant, and the car would stop instantaneously at that distance every time, regardless of velocity. Unfortunately, we live in a world of latency, so it takes time for messages to travel from one point to another. More specifically, a non-negligible amount of time elapses between the instant the car crosses the stopping threshold and the instant the motors stop moving. During this time, the car continues to move at a given velocity. Therefore, for higher velocities, the stopping threshold should be higher. The exact relationship between velocity and stopping threshold was determined experimentally by manually adjusting the stopping threshold; for a certain set of velocity values, we recorded the minimum stopping threshold that would prevent a crash. Once we had these data points, we applied a simple kinematic equation to find the best fit:

$$v_f^2 = v_i^2 + 2a\Delta x \tag{5}$$

According to this equation, the stopping threshold should be a function of the velocity squared. Therefore, we used quadratic regression across our data points to calculate the stopping thresholds for a set of possible velocities.

2.5 Overall Pipeline (AY / Figure by RS)

With the above problems addressed, we now summarize our overall pipeline (Figure 11) . Note that the safety controller is ready at all times, and when triggered will override any command.

1. Receive the LiDAR range scan and choose two subsets to form the left and right scans defined by the range $\Delta\theta$, indicating how far above the horizontal to read for each side.
2. Send scans into the two control pipelines A and B which output error signals ω_A and ω_B, respectively.
3. Check if there is a possible inside turn. If so, ignore the control outputs and max out the steering angle. Otherwise, continue with PID.
4. Define the overall control output as the convex combination $\omega = \alpha\omega_A + (1-\alpha)\omega_B$, where α is the Ragulan Coefficient.
5. Send ω to the PID controller and stand by for the next scan.

3 Experimental Evaluation

3.1 Making a Decision on a Validation Metric (OM)

In order to validate the system, we sought a useful metric to quantify its success. As the desired distance to the wall is used as the set-point for the controller, we decided that the error from this set-point was a relevant and encompassing value to use to evaluate the system. This error was measured from the linear regression model of the scanned wall, which we realize could lead to minor inaccuracies in some test cases. However, using the shortest distance from the actual wall could lead to inability to analyze the trends due to excessive noise, and the estimated regression line should serve as a reasonable approximation. We apply this error metric in order to tune hyper-parameters and evaluate the performance of our system.

3.2 Determining Relative Controller Weights (OM)

With the controller comprised of the velocity dependent portion and the simple PID portion, our first priority was determining the relative weight of each component that would minimize the error from the set-point. We consequently designed a test to confidently ascertain the best values for these weights. For the entirety of the experiment, the car's velocity was held at 1 meter/second, and the racecar was placed 1 meter away from a straight wall with few irregularities. The desired distance was set to 0.5 meters, resulting in an offset of 0.5 meters from the set-point. The wall follower node was then run on the racecar for approximately 10 seconds, with the error from the set-point being recorded at a frequency of approximately 10 Hz. The independent variable was the Ragulan Coefficient, which was manipulated on a range of [0, 1.0], with a higher density of trials recorded around 0.8 (our original qualitative approximation). The integral time-weighted absolute error was calculated for each trial, which we approximate by our discrete observations:

$$\frac{1}{t_f - t_0}\int_{t_0}^{t_f} |d(t) - \delta| \; dt \approx \frac{1}{t_f - t_0}\sum_{i=0}^{N} |\hat{d}(i) - \delta|\Delta t_i, \tag{6}$$

where Δt_i denotes the time between successive observations. For the lower Ragulan Coefficients, and therefore higher dependence on pure PID control, there was higher integral error. This can be attributed to over-corrections in the PID control resulting in oscillations around the set-point. Since the testing environment is a straight wall with little turning required aside from the initial correction to the set-point, the trend

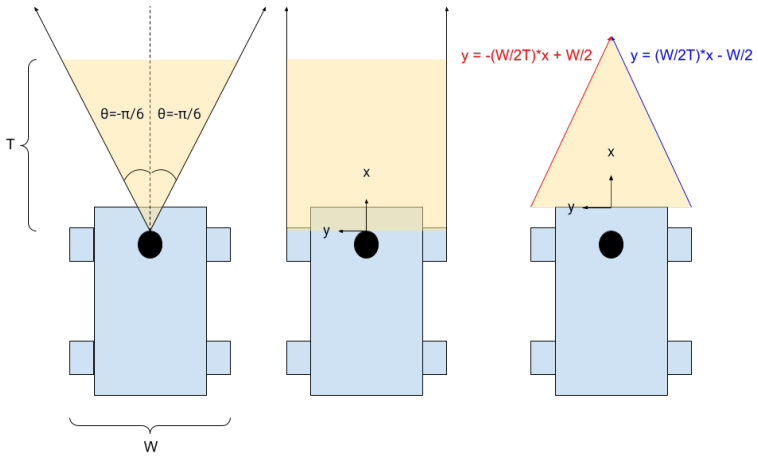


Figure 10: Iteration of our safety controller scans over time from left to right: 10.1, 10.2, 10.3. The LiDAR scanner is depicted as a black dot, and the scanning area is depicted as the region highlighted in yellow. Variables: d = stopping threshold, W = width of the car.

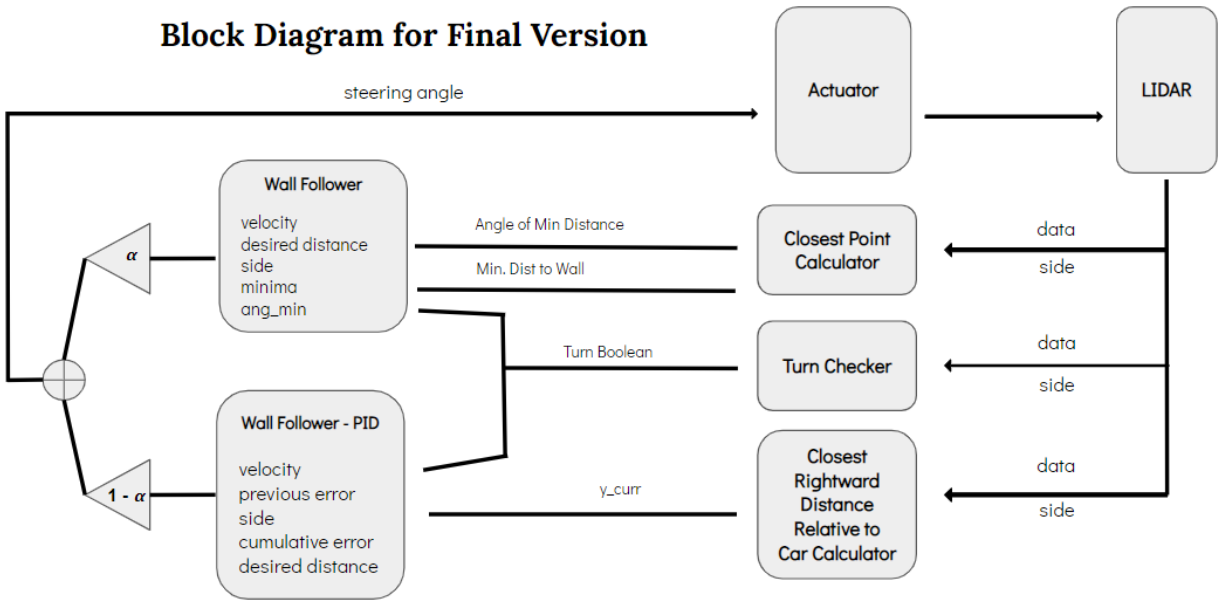


Figure 11: Block diagram outlining the overall pipeline. Note the role of the safety controller as a supervising node.

line predicts that the higher the Ragulan Coefficient, the lower the integral error; however, we observed that depending fully on this velocity dependent component reduced the reliability of outside turns. Ultimately, we combined these observations to land on a value of 0.82, the best observed value after taking the turning trade-off into account.

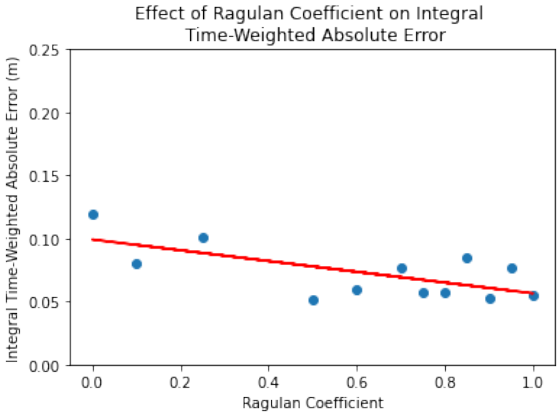


Figure 12: A graph illustrating the relationship between the Ragulan Coefficient and the integral time-weighted absolute error of the racecar from the set-point. The conditions include a 10 second interval on a straight wall with a set-point of $\delta = 0.5$ meters, and initial distance from wall of 1 meter.

3.3 Determining a Model for Safety Distance (OM)

Through qualitative testing of the safety controller and application of basic kinematics principles, we determined that there needed to be some velocity dependence for the safety controller stopping distance. Higher speeds required a higher stopping distance to allow more time for the car to decelerate to 0 velocity. Since it was difficult to measure the deceleration of the car by hand, we decided to collect data on the distance required for a successful stop for a range of speeds [0.5 m/s, 3 m/s], and analyze these values to create a generalized predictor function. The predictor function, visualized in with the lowest coefficient of determination ($R^2 = 0.99081$), and therefore the best fit, was second order. This is consistent with the physical explanation described in section 2.4.

Table 1: Safety Distance at Different Velocities

Velocity m/s	**Safety Distance** m
.5	.30
.75	.45
1.00	.55
1.25	.65
1.50	.72
1.75	.79
2.00	.87
2.25	1.02
2.50	1.35

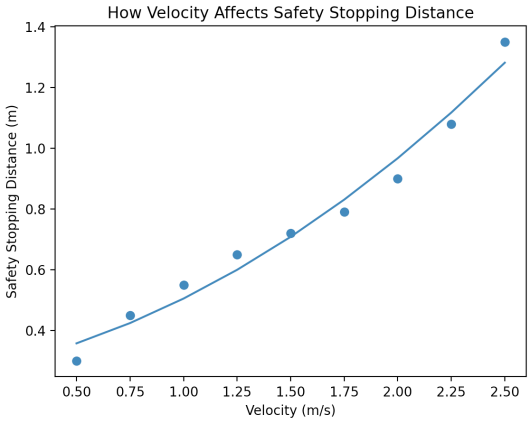


Figure 13: A graph illustrating the relationship between velocity and stopping distance for the safety controller. We get $y_{fit} = 0.1113x^2 + 0.1280x + 0.2656$.

While this model is representative in theory, we sought to validate the reliability of this model. We devised three classes of obstacles that the car should be able to detect and not collide with. The first was a static obstacle, which we represented using the brick from the lab materials (Figure 14). In application, any obstacle lying in the path of the robot without changing its position would be a static obstacle. We also tested for obstacles with horizontal velocity, represented by a walking group member (Figure 16), and obstacles with vertical velocity, represented by a falling brick (Figure 15). Based on these three expectations, we created a small test suite that should create an illustrative picture of the capabilities of the safety controller. These tests were analyzed qualitatively, as the success of a safety controller is binary. As depicted, all three of these tests were successes indicating a high level of robustness.

Figure 14: A static obstacle safety test using the stationary brick.

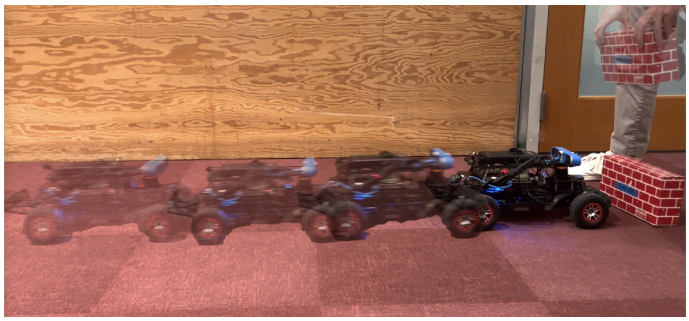


Figure 15: A vertically moving obstacle safety test.

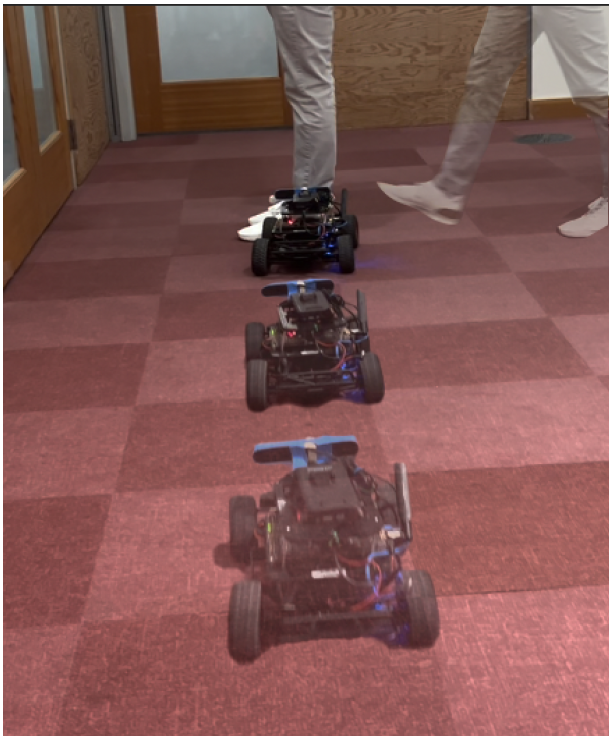


Figure 16: A horizontally moving obstacle safety test.

3.4 Evaluation of Final System (SY; Plots by AY & OM)

In this part of the report, we will the experimentally test the final wall-following algorithm, quantitatively and qualitatively evaluating the strategy's performance. Our evaluation consisted of a test suite of six scenarios, designed to test the algorithm's capabilities in following straight walls, navigating turns, and completing complex environments. For each scenario, we compare the performance of the initial PD design to the final design, which incorporates a convex combination of the two control pipelines and a Ragulan coefficient of 0.82 that we hypothesized would improve the stability and robustness of the algorithm. Overall results are summarized in Table 2.

Table 2: Overall Test Results (Average Error) Across Six Scenarios

Test Scenario	1	2	3	4	5	6
MAJORS	0.057430	0.023599	0.03827	0.13222	0.33494	0.20890
PID	0.11907	0.063565	0.06730	Fail	Fail	Fail

Scenario 1: Straight Wall Follower with No Perturbation

The first scenario involves a straight wall follower with no perturbation. This test is designed to assess the algorithm's ability to follow a straight wall without any external disturbances. In this scenario, we set the Ragulan coefficient to 0.82 and compared the performance with the initial PD design, where the coefficient was set to 0. To ensure a thorough evaluation, we set the initial error to 0.5, a value that is high enough for the algorithms to show their convergence properties to a lower error value.

The plot in Figure 17 shows the error values for both designs over time, "MAJORS" indicates the Ragulan 0.82 implementation. As we can see, the curves shows that the final "MAJORS" design reaches an error of 0 much faster and has less oscillation. This suggests that the inclusion of the Ragulan coefficient in the control error mix improves the stability and robustness of the algorithm in following a straight wall without any perturbation.

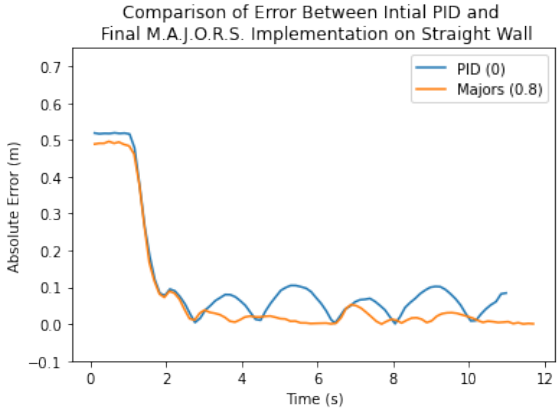


Figure 17: A graph depicting the improvement from the original PID design to the final M.A.J.O.R.S. design for the case of a straight wall with no initial perturbation from the set-point.

Scenario 2 & 3: Straight Wall with a 30 Degree Angle In and Out

For scenarios 2 and 3, we tested the algorithm's ability to handle external perturbations in the form of a 30-degree angle in and out of the straight wall, respectively. As shown in the diagram of the test setup Figure 18, the wall was positioned with the angle in or out to create the desired perturbation.

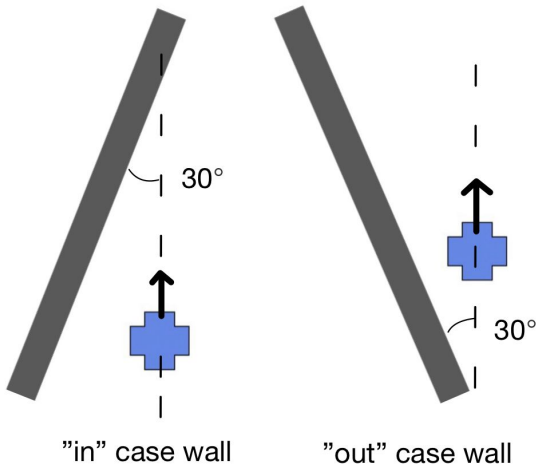


Figure 18: An illustration of “in” and “out” wall case setups. In the diagram, the car follows the left wall. For the “in” case, the wall leans towards the center axis of the car. For the “out” case, the wall leans away from the center axis of the car.

In the “in” case, we set the initial error to 0.1. This value was chosen because a smaller wall-following distance requires the algorithm to have higher recognition and response capabilities. In the “out” case, we set the initial error to 0.25. This value was chosen because in this case, the algorithm needs to maintain robustness to prevent loss of following ability from where the wall is in a direction away from the vehicle.

The test result Table 2 shows the cumulative mean error values on time for both designs in the “in” and “out” cases, MAJORS designs both have lower mean error than just PID. The plots in Figures 19 and 20 further illustrate this, MAJORS has a consistently lower error values, showing that the final design has less oscillation and reaches an error of 0 faster than the initial design. This indicates Ragulan coefficient mixed control error improves the robustness of the algorithm in following a straight wall with angles.

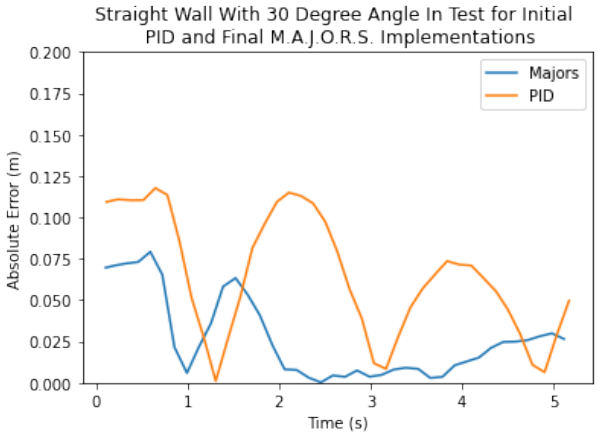


Figure 19: A graph depicting the improvement from the initial PID design to the final M.A.J.O.R.S. design for the case of a straight wall with the car angled *in* by 30° to start.

Scenario 4 and 5: Testing the algorithm’s ability to handle inner and outer corners with a 90-degree angle

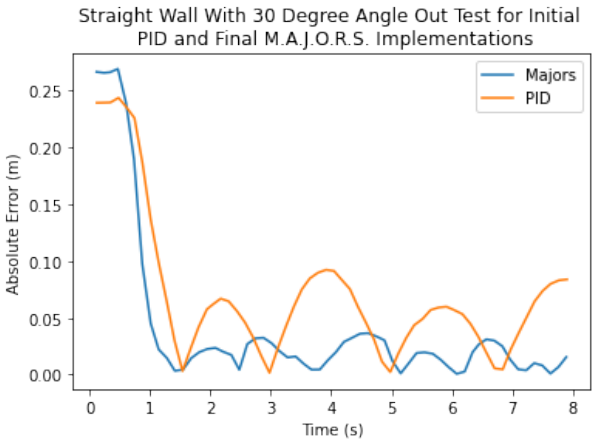
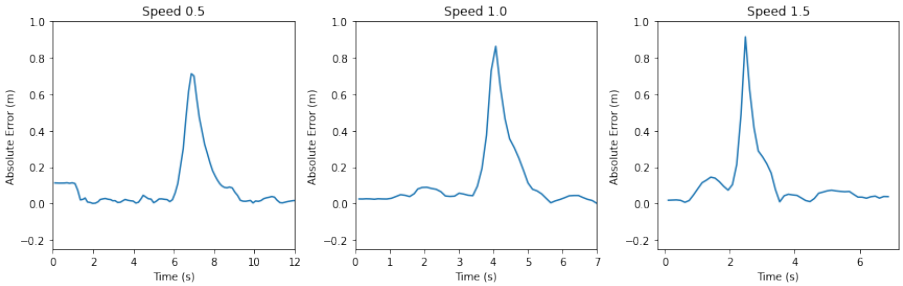


Figure 20: A graph depicting the absolute error over time for the initial PID and final M.A.J.O.R.S. implementation for a straight wall test with the car angled *out* by 30° to start

Scenario 4 and 5 were designed to test the algorithm's ability to detect and follow a second wall at a 90-degree angle. In Scenario 4, the car started on the inner side of the corner, while in Scenario 5, the car started on the outer side of the corner. To achieve this test, we created a test course with a straight wall and a 90-degree corner with the second wall as Figure 6. The car with Ragulan coefficient of 0.82 started at the beginning of the straight wall and was required to follow the wall around the corner and then follow the second straight wall.

Figure 21 and Figure 22 illustrate the error over time for the final design with the Ragulan coefficient of 0.8. The plots show that the error value increases rapidly and decreases rapidly, indicating a fast response time of the algorithm when detecting and following the second wall. Also, the peak value of oscillation during following is lower than that of the initial PD design in straight falling, indicating that the algorithm has a more stable and robust response when encountering a corner.

Additionally, for the "in" case, we tested several speeds, including 0.5, 1.0, 1.5, 1.75, and 2.0. The purpose was to evaluate the suitability of the algorithm for different speeds. The results showed that the final design worked well for all tested speeds, with no signs of failure or collision with the wall. In the "out" case, we tested the algorithm's ability to handle a typical velocity of 1.0. The results showed that the final design can efficient detecting and following of the wall.



Scenario 6: Around the world. Testing the algorithm's ability to handle a complex room

In this scenario, we tested the algorithm's ability to handle a complex room with multiple walls and corners. To achieve this, we designed a test course with a complex room that included multiple corners and walls, making wall-following tests around a room with complex layouts.

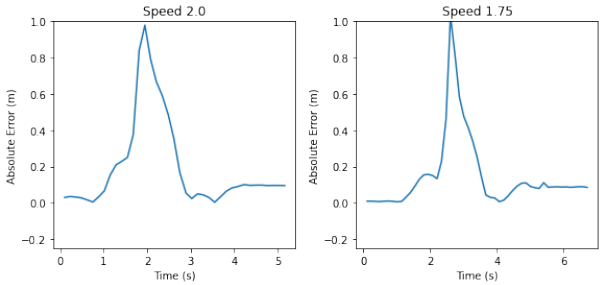


Figure 21: A collection of graphs depicting the absolute error over time for an inner turn for the M.A.J.O.R.S. design at varied speeds (m/s). Not pictured are the results of the original PID design, as it failed the 90° outside turn that was used for the test case at every speed.

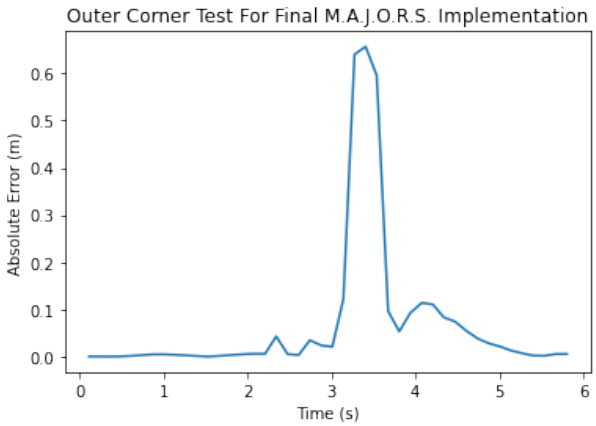


Figure 22: A graph depicting the absolute error over time for the final M.A.J.O.R.S. implementation for an outer corner test.

We compared the performance of the final design with the initial PD design. The plot in Figure 23 illustrates the error over time for the both designs. It shows that the initial PD design failed to follow the wall after it encountered the third corner, as seen in the curve that at the third peak of error, the curve stays at a high value and didn't come back to 0. However, for the final design with the Ragulan coefficient of 0.8, the car successfully worked until the end without any failure in any corner. Also, the peak value of oscillation during following is lower than that of the initial PD design, indicating that the algorithm has a more stable and robust response when encountering corners.

Overall, these scenarios results suggest that incorporating the Ragulan coefficient into the control system can significantly improve the wall-following algorithm's performance, particularly in complex environments with multiple corners and wall directions. The mixed control error of PD and Ragulan parameters allows for a more adaptive response to varying conditions and enhances the algorithm's ability to navigate through challenging terrain.

4 Conclusion (MW / edited by OM)

From the outset, the group's goal was to develop a capable and adaptable wall following algorithm for an autonomous racecar. The algorithm subscribes to the LiDAR scan and processes it through linear regression to model the wall's position. It then publishes a steering command based on the error from the wall and the car's velocity using an augmented PID controller that intends to minimize this error. The car is also

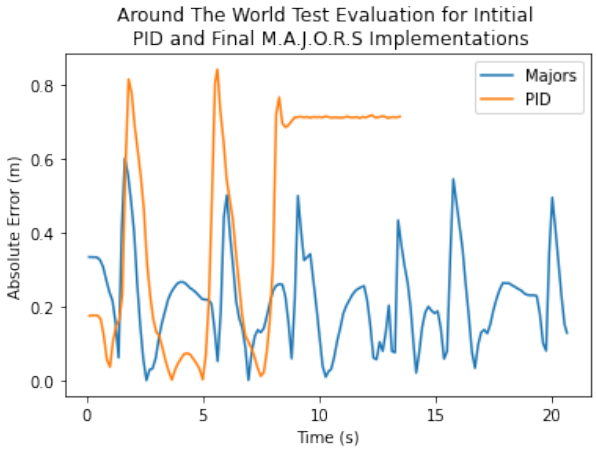


Figure 23: A graph depicting the absolute error over time for the initial PID and final M.A.J.O.R.S. implementation for an "around the world" test, which was comprised of six 90° inner turns around a room.

equipped with a safety controller that uses LiDAR data to identify unexpected obstacles around the car and brake the car smoothly to prevent collisions.

We overcame challenges such as minimizing oscillations and effectively detecting and navigating corners. Our group's combined efforts resulted in a more robust controller that performs well in both simulated and real-world settings. The safety controller works in the background, continuously monitoring the car's movement and taking over autonomous control if necessary, which adds an extra layer of protection to the racecar.

We are aware of the limitations that still need to be addressed in future labs to improve our racecar. One of the issues we encountered was that the car is unable to detect obstacles that are low to the ground due to the placement of the LiDAR. Additionally, reflective surfaces introduced a degree of confusion in the LiDAR data, hindering the car's performance. We also acknowledge that there is a ceiling on how sharp of a turn the car can take due to its turning radius. Another limitation was that the car can only take into account one side at a time (left or right). Furthermore, we found that the safety controller is somewhat unreliable for speeds in the range of 3 to 4 m/s. We also found that we had too many hyperparameters to tune, making the tuning process inefficient and time-consuming. This affected the overall performance of the racecar and limited its ability to navigate and perform tasks effectively. However, as a team, we are committed to finding ways to optimize the tuning process and improve the racecar's performance in future labs. Overall, we worked cohesively and are looking forward to the next lab where we can continue to develop our racecar's abilities through computer vision.

5 Lessons Learned

Michael During the lab, I gained knowledge in both technical and communication aspects. Among the challenges we faced, I found the processing of LiDAR data and tuning of the PID controller to be particularly difficult. In retrospect, I should have been more proactive in using my expertise in dynamics and controls to solve these issues earlier. It's important for us to recognize each other's strengths and utilize them appropriately to succeed in this class. This was also my first experience collaborating on code, and I learned the significance of communicating my thought process to my teammates. As we have now become familiar with the hardware, I believe that delegating tasks and planning our workflow more effectively will not only help us solve technical problems faster but also improve our briefing and report organization.

Alan From a technical perspective, this lab introduced me to PID control and taught me how to think critically about car kinematics. A highlight of my experience working with the PID controller was thinking

about making the control a weighted sum of two different error signals (using the Ragulan Coefficient). For car kinematics, I enjoyed thinking about the geometry of the inside turning problem when our original regression approach did not work out. I also learned there is a balance between tuning hyperparameters and implementing more logic for corner cases. For example, tuning the gains can be very helpful to improve turning, but only to a certain extent. It was really helpful to implement separate turning logic in this case. From a CI point of view, this lab made me realize how important planning, organization, and communication is to success of a team. I made a Notion page for the team which we use to hold important documentation (e.g. how to start the car), task lists, meeting notes, and important links. It helped a lot to document our daily work and have a home base to jot down important information. Planning was essential for our testing phase in order to not get lost running arbitrary trials, so we prepared a list of test cases and a test suite for overall evaluation. Still, there are many areas for improvement here. To make the next lab smoother, we should utilize the task list more and set deadlines and milestones for each person in order to hold each other accountable.

Josh Technical: First, I learned about the capabilities, but more importantly the limitations, of PID controllers. Through 2.003 and 2.004, I have a solid theoretical background in controls. However, this was my first time coding my own PID controller from scratch. I've also never combined different controllers before, so optimizing two controllers to work simultaneously was an interesting problem to solve. Second, I experienced the frustrations of the sim2real gap firsthand. All of our controllers performed very well in the simulations (around 97%). However, none of our controllers actually worked in real life. For instance, my controller had code that explicitly scanned the front of the car and turned accordingly, which worked nearly perfectly in the simulation. In real life, my controller was completely unable to perform inside turns, instead opting to run directly into the wall. I'm still not entirely sure why this occurred, since we didn't end up using my controller for our final code, but this ordeal made me very aware of the existence of the sim2real gap. Communication: I learned that there is truly no replacement for in-person meetings. However, with a team of 6 busy MIT students, it's impossible that we will always be free at the same time, especially as the semester ramps up. To solve this issue, we started each meeting by de-briefing any new developments to catch up those who missed the last meeting. To this end, meeting notes were extremely helpful to document all major design decisions. Going forward, the Notion that Alan made will become even important as a central document of our progress as a team over time.

Owen The technical portion of the lab was my first experience working with PID control, as well as my first time writing code that would be dictating the actions of a hardware system. My first takeaway was that in order to avoid wasting precious time, the group had to establish an efficient workflow for setting up the car and sharing code. We created a GitHub page where we could push/pull code, as well as separate branches with each of our implementations. As we ran into various hardware issues, we found it valuable to stay level-headed and focus on the work we could do without a working car. Solving these hardware issues took time, but also allowed us to quickly recognize and fix reoccurring issues in the future. I particularly learned for the testing process, which I spent a good portion of my time on. We initially found ourselves constantly manipulating parameters and constants without much rhyme or reason. I've realized that if we find ourselves doing this, the best option is to take a step back and reevaluate the current system. For example, we wasted a significant amount of time attempting to tune the proportional constant for our PID controller with various tests, but to no avail. This led us to try integrating Ragulan's code into the controller, and we saw improvement instantly. It can be dangerous to get stuck in one thought pattern when hitting a roadblock, and I hope to apply this lesson to future labs and my career at large.
One of our first objectives as a team was to establish a mode for sharing ideas. We created a shared Notion AI page for this purpose and used it for posting data, linking videos/pictures, recording meeting notes, and attaching important pages like the car sign-up spreadsheet. I think all of the team members found this very useful, and we will continue to use it for the remainder of the semester. One issue that I believe we can improve upon is delegation of time. We were slightly disorganized with meeting times, and were hardly ever able to get the whole group together, except for lab hours. In the future, we will be using a when-to-meet form to determine everyone's availability in order to schedule meeting times.

Ragulan On a technical front, this lab taught me about the limitations of theoretical designs for real-world devices. I spent lots of time trying to find the optimal steering angle, and I implemented it with miniscule error in the simulation. However, when put onto the racecar, problems were immediately apparent. Certain ranges of my scan data were polluting my algorithm and made my car spin in a circle. After I fixed those, my racecar still oscillated about the desired distance when following walls because it constantly overshot the desired distance. That is to say nothing of the issues I encountered with turning. None of these problems were present in my simulation, and this showed me the limitation of simulation: the real-world isn't ideal, and we must account for this in our designs and simulations, for otherwise we'll not see the error until deployment and have to spend hours debugging.

From an organizational and teamwork frame, this lab taught me the value of starting early, having a plan, and being in constant communication with my teammates. I have friends in other groups, and I am far less stressed than them, and it's because we started early and sustained that work ethic throughout. We had a timeline to finish early, and we always had actionables to complete before we next met together. This made sure we never fell behind and never felt lost. Also, with us always being in communication over text, it's so easy to get help from a teammate and re-oriented onto the right path. When I had the racecar on one of the first nights, I couldn't figure out how to run my wall-follower code, but a short five minutes later, I was able to, for I got an immediate response from my teammates on what I was doing wrong.

Going forward, we should make outlines for labs as a whole to start and assign tasks with a clear plan of action for each night. We had meetings at the beginning without progress, and this was because our tasks didn't have a clear plan of attack and no one could make meaningful progress. Addressing that will increase our productivity and make for even easier times in the future.

Shenzhe From the technical perspective, during the lab project, I gained a deeper understanding of PID controllers and their applications in robotics. Specifically, I learned about the importance of selecting the appropriate combination of error signals to achieve optimal performance. Through discussions with my team members, we realized that the different error signals reflected different features of the system, and that combining them in different ways could help us to achieve our performance goals. As in the final implementation, we set the Ragulan coefficient to help us take the y-coordinate of the closest point into decision making consideration, and end up with a great performance. On experimental side, we tuned the PID gains in real world to achieve the desired performance of the system by analysis of the overshoot, oscillation, settling time. During tuning, we understood that the choice of error parameter selection method was not simply a matter of trial and error, but rather reflected a deeper understanding of the underlying physics and dynamics of the system. By analyzing the response curve, my team and I finally determine that a Ragulan Coefficient of 0.82 provided the best performance for the system, as it better reflected the vehicle's ability to respond to complex environments.

Also, throughout the lab project, our team utilized a number of tools to improve our communication and workflow. We have a shared GitHub repository, Notion site for tasks arrangement. These tools were invaluable in enabling us to work effectively and efficiently as a team. Also, at the beginning of the project, by summarizing and absorbing everyone's highlights in the former project, we were able to build a well-thought-out wall follower implementation that incorporated the best practices and insights from each team member. We should continue to use these way to improve the efficiency and comprehensiveness of group projects.