Lab 3 Report: Effectiveness of Dual Mode Driving Controller and Bounding Box Safety Controller in Autonomous Wall Following

Team 6

Penny Brant
Yatin Chandar
Fritzgerald Duvigneaud
Nico van Wijk
6.4200: RSS

March 11, 2023

1 Introduction

The objective for this lab is to create a controller for autonomous vehicles to accurately follow a wall safely and at a distance using LiDAR data. Incorporated into this goal is the development of an accompanying safety controller in the event of an unexpected obstacle appearing.

Wall following is a common challenge in autonomous driving that relates to many applications such as ensuring safety of passengers in self-driving cars or path following. More specifically, being able to perceive and react accurately remains a key issue for all aspects of autonomous driving. Thus, the principles utilized in this lab will be key building blocks for future challenges as we move forward with the development of this project.

In this report we demonstrate that we are able to achieve the creation of a robust wall following controller through the iterated development of multiple algorithmic strategies. Ultimately, by developing responses to the flaws of each of our strategies, we settled on a dual mode wall following controller that uses Hough Transform estimates to map the walls of its surrounding environment and relies on both angle and distance to the nearest wall to response plan. We found this solution to be effective across a range of driving cases and comparatively much safer than our earlier attempts with a simple single proportional–integral–derivative (PID) controller based entirely on distance to the desired side wall. Furthermore, we achieved our incorporated goal of a safety

controller by developing a bounding box region in front of the car that will sent halt commands if objects occupy a last enough portion of it's region to be a threat to the vehicle.

2 Technical Approach

2.1 Overview

Our final technical solution that we settled on for this lab utilises Hough Transforms to convert the LiDAR sensor data into estimates of all surrounding wall locations. We then use the distance and/or angle from the desired side wall that we want the car to follow and feed an error derived from the difference between this distance/angle the desired distance/angle into a dual mode controller that depending on the state of the car, will steer it using one of two PID controllers configured with different gains. The states we use at this stage are rather simple and are basically whether the car is currently aligned to and following the wall, or facing in an incorrect direction/in open space.

The other state that we consider is the one where the race car is approaching a corner. When the car is an appropriate distance from the corner based upon front LiDAR data, it will initiate a hard turn. We did attempt to create smooth arcs to approximate walls and "corner cut", although the regular controller to just follow this simulated path instead however this approach was never robust. The algorithm attempted for this, as well as our general design steps, iteration process and intermediate solutions we trialled are explained in the rest of this section.

2.2 PID Controller With Forward Looking

Our initial implementation of the wall following algorithm was implemented upon one of our team's members solutions for lab 2: simulated wall following. This solution utilized a proportional–derivative (PD) controller that relied on the distance between the wall directly to the desired side and the desired distance for its error signal. At low speeds (0.5mph), this controller with low gains ($K_D = 0.1$, $K_p = 0.2$) was effective at minimizing the error and keeping the racecar close to the wall. However we quickly realized that the gains should ideally velocity dependent as the car was less responsive at higher velocities at the same gain levels. We eventually introduced an integration term and recalibrated the gains to improve this controller, however it still never became reliable across the car's full velocity range.

This algorithm also relied on a simple look ahead scan to determine approaching corners. By taking the 5 closest LiDAR scan points directly ahead of the car and averaging their distance, the car would initiate a hard turn at maximum steering

angle when this distance was less than a certain threshold. Although the initial version of this code from the simulated lab had defined a threshold based upon desired distance, we also quickly realized this threshold was also a function of velocity. Through trial and error, we eventually defined a relationship between corner detection threshold distance (d) and velocity (v) as:

$$d = 1.5 \cdot \sqrt{v} \tag{1}$$

This function seemed optimal at a speed of around 2mph but although the race-car successfully managed to go around corners at other speeds, it was far from smooth.

Lastly, regarding the PID approach, we realized that this model was flawed in a number of edge cases - two of which are demonstrated in the sketch in figure-1. Firstly, if there was an obstacle that extended from the wall on the desired side but protruded less than the desired distance, it would be possible for the car's wheel or front bumper to hit said obstacle before any of our sensor slices picked up on the obstacle. Secondly, if the car was placed in open space, it could often spiral and spend a long time finding the nearest wall. Thirdly, if there was an inside turn on the wall that the car was following, it was slow to react and could possibly miss the turn entirely if it was like a doorway and the wall continued past it. These edge cases are not exhaustive but do demonstrate flaws in the design.

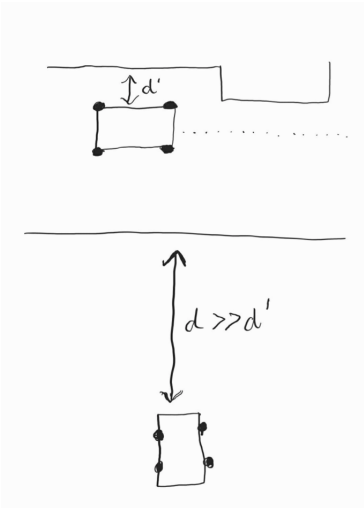

Figure 1: Sketches of edge cases not handled by our initial PID Detection. Top: object protruding from the wall less than distance d'. Bottom: car starting in open space not near a wall

2.3 Dual Mode Controller

To address some of our edge cases concerns above, we next devised a dual mode controller. Rather than have a singular error condition (distance from the desired distance), we instead decided to also calculate the angle of the car towards the closest wall. When the car was far away from the wall, the controller used the distance from the wall as its error. Since wall distance is a value that changes quickly with steering angle and speed, this controller mode was more aggressive and quickly moved the car to approximately the target distance from the wall. However, since steering, speed, and perpendicular distance from the wall are all dependent on each other, maintaining the target distance from the wall using just this controller was very difficult. This is where the second controller mode activates. If the car is less than 0.1 meters from the wall, the program will attempt to correct the angle of the car relative to the wall, minimizing drift. The angle controller has no distance control, only trying to keep parallel to the wall by using the inverse tangent of the slope of the detected wall line as the error. The inverse tangent of the slope is the angle of the wall relative to the car, and if this angle is minimized, the car remains parallel to the wall. If the car's wheel geometry starts to make the car drift out of parallel with the wall, the distance from the wall will exceed the threshold, and the aggressive distance controller will take over.

2.4 Hough Transform for Wall Detection

An outstanding key challenge that remained was finding accurate and meaningful segmentation to detect and differentiate side and front walls. In our previous iteration of the controller, we segmented the LiDAR data by angles and used linear regression to estimate wall locations, but this led to inconsistent perception of walls when the angle of segmentation didn't line up with the corner. To improve upon this, we implemented the Hough Transform. This technique converts a set of ordered points into clusters that represent lines using two parameters: R, which represents the closest distance from the line to the origin, and theta, which is the angle between the line and the origin.

We show a sample Hough transform for data with no noise in figure-2 below. It transforms the data into 3 points corresponding to the first line segment, the second line, and the point of discontinuity.

In figure-3, we show the Hough transform on generated data that contained noise, where we can clearly see that the clusters for Hough transform are hard to see even in this generated data with limited noise.
To identify clusters of data quickly and eliminate noise, we down sampled the data by using a sliding window averaging the n data points before applying the Hough Transform. Then, we classified lines that had an angle between them less than 30 degrees as the same line. In figure-4, we illustrate the effect of this

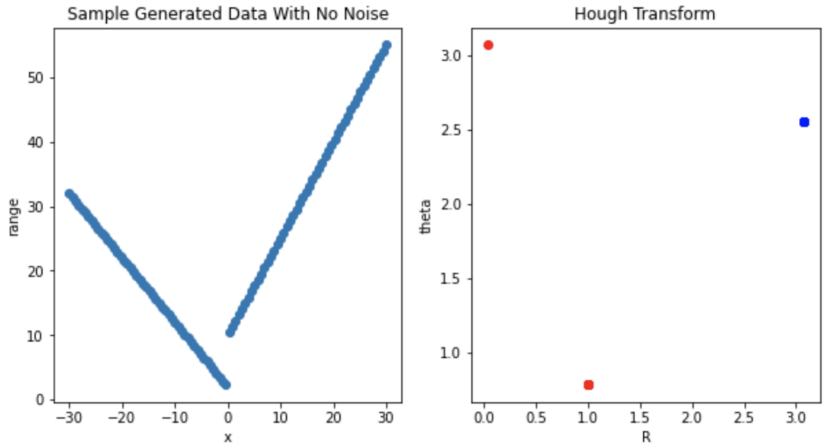

Figure 2: Hough transform on example lines with no noise

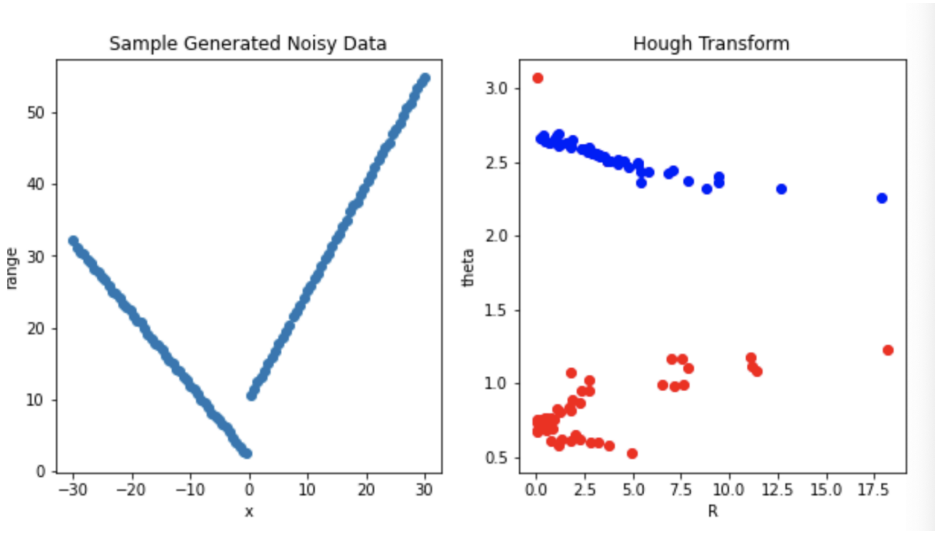

Figure 3: Hough transform for noisy data with no downsampling, where the two lines are represented by different colors

smoothed Hough Transform when we averaged every 10 points.

Using this method, we were able to identify meaningful lines based on the full range of LiDAR data, as illustrated in figure-5. In practice, we used the smoothed Hough Transform algorithm with a downsampling of 50 data points, as we found that anything smaller than it caused lagging in the robot, which led to bigger calculation errors down the line.

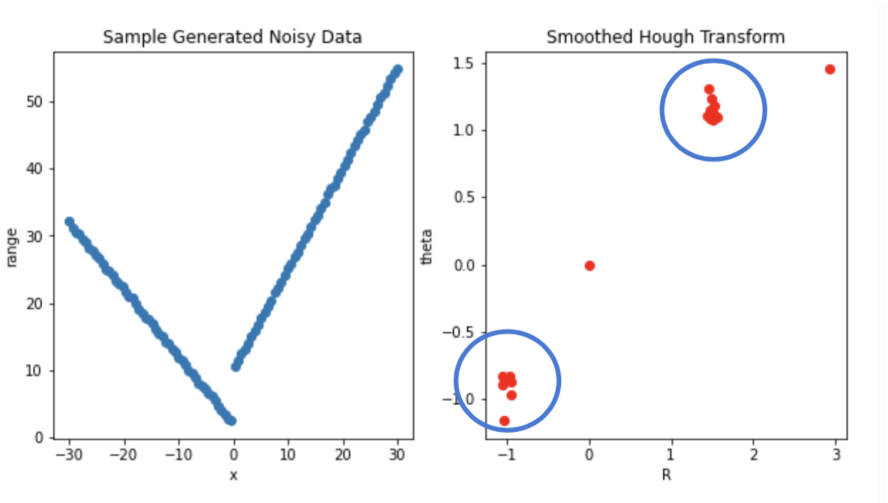

Figure 4: Hough transform for sample generated noisy data where data is down-sampled to avoid noise.

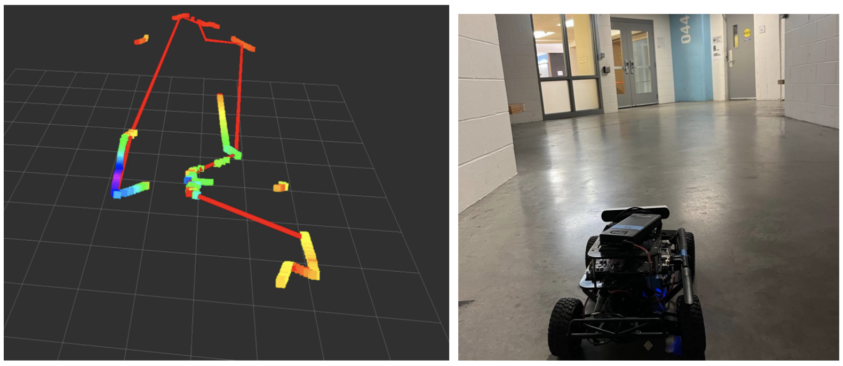

Figure 5: Wall segmentation based on LiDAR data using smoothed Hough transform and error rate to determine lines baked on, here the LiDAR is pointing away from the camera.

2.5 Wall Segmentation

For wall segmentation, we explored various methods for wall segmentation using clustering, error margin, and progressive recursion. However, many of these methods ran into latency issues, which led to errors reliably selecting the wall.

Initially, we implemented both K-means and agglomorative clustering to attempt to group lines with a similar slope together. However, using these clustering algorithms showed that the Hough transform was extremely sensitive to noise, with the cluster quickly becoming merged as the car moved far from the wall, or a front facing wall was approaching the car. Future work will depend upon using the probabilistic Hough transform, which is more robust and has lower computational overhead. For this application, we utilized the inherent knowledge that the laser scan data was ordered to perform progressive recursion to determine the locations and slopes of straight lines in the laser scan data. Progressive recursion relies on the principle of the R^2 score. This score is minimized when performing linear regression, and is a good way to assess the fit of the regressed line. The algorithm starts by performing a linear regression over the entire laser scan dataset, This results in a fit with a very poor R^2 score, so the program recursively reduces the size of the data set by removing a data points until the R^2 score rises past a cutoff - in this case, .99. This cutoff indicates that the regressed line fits the points very well, meaning that the series of points used to perform the regression must be in a straight line. The algorithm segments off this data, and performs the same calculation on the rest of the data, iterative finding all the straight lines in the set of scan points. The main benefit of this algorithm is that it will always find the largest groups of points that are in a straight line, which results in few iterations. However, the drawback is that performing the large number of trial regressions required is computationally expensive, and led to publishing latency and dropped frames. The concept of progressive regression is very interesting, and we would like to explore it more in the future, with optimization and faster hardware. For this application however, we pivoted to applying downsampling to the laser scan data. This was previously useful for decreasing the computational cost of progressive recursion, and turned out to be a powerful method for determining straight lines by itself. Downsampling works by taking every nth data point and creating effectively a very low fidelity LiDAR. This almost completely eliminates noise caused by bumpy walls, and can be easily tuned to increase or decrease fidelity. This method also avoids recursion, as the equation of the line segments that connect subsequent sampled datapoints can be used as the segmented walls. We think that this can be an effective way of redicing computational load while retaining enough fidelity to detect large features such as walls and corners, and even follow curved walls. A dynamic downsampling perception algorithm is planned for future iterations due to its effectiveness.

2.6 Corner Cut Algorithm

Despite the improvements in our car's model of the surrounding walls, our look-ahead corner detection was still far too sensitive to change in velocity. To resolve this, we used our new wall representation to develop an algorithm to smooth car trajectories around corners so that our controller had an easy to track reference signal throughout the duration of the turn. Furthermore this algorithm also allows for choosing custom turning radii.

figure-6 visualizes this algorithm in "practice". The orange and blue lines define the walls of the corner. The red dot is the turn's center of rotation. The green dots demarcate the arc over which the turn takes place. The black dot is the car's current position. The algorithm computes the distance error between the black dot's radial distance from the center of rotation and the target distance. It then feeds this error signal into the distance controller which adjusts the car's heading to correct.

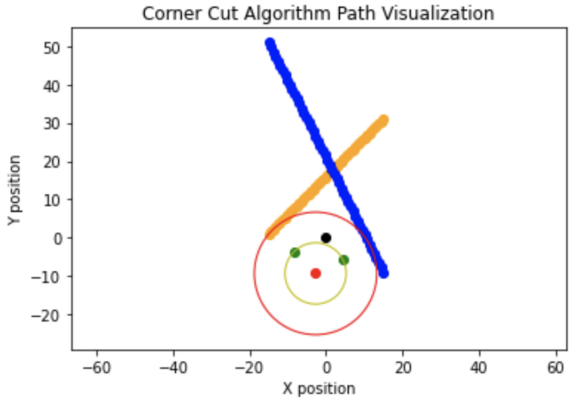

Figure 6: A visualization of the corner cutting algorithm showing the turn radii for a corner

We denote the desired distance of the car from the wall as d_t, and to avoid the car crashing into a corner, this algorithm constructs a circle with radius $r_2 = 2 \cdot d_t$ in between the two lines representing the two walls at a corner. Here, we denote the two walls using the equations $y_1 = m_1 x + b_1$ and $y_2 = m_2 x + b_2$. To construct this circle, we offset both lines by a distance $r_2 = 2 \cdot d_t$ in the direction of their normal (which is ±[-m, 1] depending on the wall following side) towards the car. After this, the intersection of the two transformed lines marks the center of this circle, and we get (x_r, y_r), which denotes the center of the circle.

We then want the car to follow the edge of the inscribed circle at a distance d_t, which is marked by the circle with radius $r_1 = d_t$ that shares the same center as the circle we just constructed. Since the car is intended to follow the wall in parallel, our smooth ideal path for the car should start when this circular path is parallel to the side wall and end when it's parallel to the front wall.

In practice, we used the LiDAR data to approximate the front and side wall, as well as the angular position of the car relative to the inscribed circle. We defined two phases of the turn: 1.) "Outside" being before the turn starts and after

the car is parallel to what started as the front wall. 2.) "Inside" being when the car's angular position is within the arc between the "Outside" segments. In order to get the controller to drive the car along the circular path, we defined the error signal for the "Outside" segments as $d_t - walldist(x_{car}, y_{car})$. For the inside segments, we defined the error signal as $r_2 - distance((x_{car}, y_{car}), (x_r, y_r))$

We certified the algorithm for the cases of right, acute, and obtuse corner angles, as demonstrated in figure-7 where the orange segments of the interior circle sweep the turning arc tracked by the algorithm.

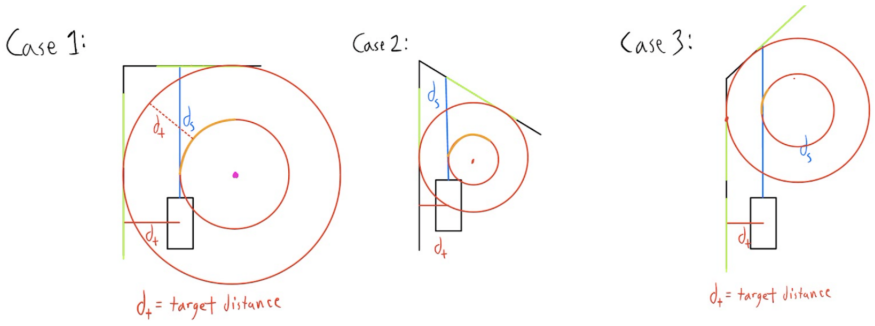

Figure 7: demonstration of the turning radii for right, acute and obtuse angles

Ultimately though, the wall perception wasn't consistent or accurate enough for Corner Cut to demonstrably outperform our previous approach of "hard turns". Because of this we ultimately chose that technique with a methodology as follows: A hard turn is defined as setting the steering angle to the maximum of 0.34 radians. We computed the corresponding turn radius of the vehicle using the formula $R_{turn} = (l_{wheelbase}/tan(\theta))$. The car has a wheelbase length of 0.33 meters. Then, we triggered the turn when the car was at a distance of $R_{turn} + d_t$ meters from the front wall. Finally we ended the turn when the car could no longer "see" a front wall.

Theoretically, this algorithm would be defeated at a corner with a sufficiently acute angle, whereas corner cut would be able to execute a proper turn. However this case did not appear in our testing and when combined with the perception issues, hard turning was the clear victor.

2.7 Bounded Box Safety Controller

Finally, we implemented a safety controller to halt the car in the case where an obstacle would appear in front of it before it could turn. Our method for achieving this was defining a bounding box in front of the car as shown in figure-8. We then calculated the percentage of LiDAR scan results that fell within this

bounding box and if it was higher than a certain detection threshold, we gave the command to the car to stop. This command was sent to a higher priority channel in the mux channel than the general drive commands from our earlier controller, hence it would override any other drive instructions.

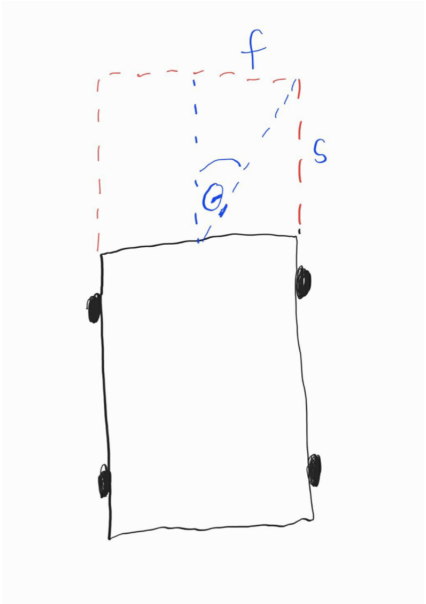

Figure 8: A sketch of the bounding box in front of the car used for the safety controller.

The method for determining whether LiDAR results fell into the bounding box was to convert the scans into Cartesian coordinates and then compare these to the parameters of the bounding box, knowing that the scanner was centered at the origin of our frame. Hence, the two relevant parameters of the bounding box as shown in figure 8 above were frame width (f) and stopping distance (s). Frame width was a constant property of the vehicle that was experimentally measured to be 0.28m. Stopping distance however was a function of velocity (v). To calculate this, we used the equation for the stopping distance of a regular car:

$$s = \frac{v^2}{2\mu g} \tag{2}$$

In this equation, g is the acceleration of gravity which is a constant here on Earth but is the coefficient of friction which is dependent on the surface that the vehicle is driving upon. To estimate our constant for the safety controller, we decided to research some common values of the friction coefficient such as

the value between rubber and concrete (0.35 - 0.45) to represent Stata basement or between rubber and polyurethane (0.2 - 0.25) to represent Johnson Track. Ultimately to err on the side of safety, we decided to use the lower bound of our research (0.2) and a detection threshold of 5% for our safety controller implementation.

2.8 Tradeoffs

In this lab, we explored various approaches on both the perception and control of the autonomous vehicle. For robot perception, we started off using simple linear regression on a interval of angles, then explored various forms of Hough Transform (both traditional and downsampled), as well as various wall segmentation algorithms like clustering and progressive recursion. On the control side, we experimented with methods of PID and corner cutting. After evaluating the latency, accuracy, and reliability of these methods, we decided to use a laser scan downsampling and duo mode controller.

3 Experimental Evaluation

3.1 Testing Procedure

In the testing procedure, we wanted to test and evaluate the performance of our algorithm on a set of commonly seen situations in wall following. We defined 3 testing tasks:

1. Running around the room at SPEED = 1
2. Straight line following for 5 meters, at speed = 2, desired distance = 0.6
3. 90 degree turn at speed = 1, desired distance = 0.6
4. 90 degree turn at speed = 2, desired distance = 0.6

Through these tests, we hope to gain a good understanding of the robot's performance under a variety of conditions and circumstances that provide a holistic and comprehensive understanding of the robot's performance. We performed these tests first in RViz and moved on to the physical robot through both qualitative and quantitative approaches. Desired distance is 0.6 for all test cases.

3.2 Evaluation Metric

To quantify our algorithm performance, we were inspired by the autograder for Lab 2: simulated wall following. Hence we designed our evaluator to use the same following scoring function:

$$score = \frac{1}{1 + (\alpha * l)^2} \tag{3}$$

Where loss (l) was the average error across the whole trial and α was a constant that we set at 1. This metric allowed us to compare both between our different controller algorithms but also between different test scenarios to identify areas of weakness in our design.

3.3 Results

Test	Description	Evaluation
1	running around the room at speed = 1	98.72 %
2	Straight Line, SPEED = 2, DESIRED DISTANCE = 0.6	99.88 %
3	90 degree turn, SPEED = 1, DESIRED DISTANCE = 0.6	98.8 %
4	90 degree turn, SPEED = 2, DESIRED DISTANCE = 0.6	99.72 %

Additionally, here's a video demonstrating our car driving around the room for qualitative assessment in this video. Qualitatively, we found that the robot performed under expectation and specified behavior under a range of different situations in terms of physical setting, speed, desired, distance, among other variables. Quantitatively, we see that the car performed well under a variety of tests in the metric we defined. Future steps we can take in terms of perfecting our perception and control algorithms may include

4 Conclusion

Ultimately, despite a number of road blocks during our technical design, we believe that we achieved the main goals of this lab. As shown during our results video, our autonomous race car successfully managed to follow walls around an entire room, including the corners. However, there is definitely still room for improvement.

The main elements of this lab that are lacking are more robust quantitative measures of evaluating our results under different conditions. Part of this is simply not conducting enough tests due to time constraints but that was also a reflection of our larger time allocation. It was also disappointing to not be able to complete the corner cutting algorithm, albeit the maths involved guided

our later distance calculation when we reverted to a look ahead approach for turning. Meanwhile, our work on Hough Transforms was encouraging as we can apply a lot of learners towards the next lab involving computer vision where the technique will be very useful.

5 Lessons Learned

Penny: In this lab, I learned valuable lessons regarding time management, programming, testing, and debugging, as well as presenting our work. I'm excited to learn more about autonomous perception and control, as well as how to communicate our works! I also discovered that it's better to finish your report before exploring innovative ideas like alternative perception algorithms with Hough Transform and Pure Pursuit (which we worked on briefly before deciding that we didn't have enough time). In this lab, everyone on team was super excited to test out new things, which was great, but it wasn't great when we tried to combine 5 innovative ideas last minute in several all nighter. In the future I think we learned our lesson to make a better balance out of being creative and practical.

Fritz: During the lab, I learned the importance of understanding the trade-off between time management and quality in engineering. While I greatly enjoyed coming up with interesting algorithms to tackle the problem at hand, I realized that it is easy to fall into a long spiral of debugging during their implementation. And just as in industry or academia, time is not unlimited. To respect mine and my teammates' time, and to meet the assignment's deadline, I had to learn to compromise on the complexity of the techniques we used.

Nico: The main personal lessons I learnt in this lab were related to general time distribution on the project. As the member of our team in charge of organising logistics such as group meetings, car work time and communication with other teams and course staff, I encountered a number of challenges. One such example was brokering an agreement for the splitting our two cars in pod 3 during lab time which took a 20 min text discussion when it would've much simpler if we established agreements from the beginning.

Other challenges included underestimating how much time the reporting and evaluation would take. Whenever our team had an early working but non ideal controller, we elected to keep iterating rather than moving onto quantifying evaluation and writing our results. This endless loop of iteration and perfectionism meant that we were forced to keep working right up the deadline and still ended up submitting a non perfect solution. Moving forward, our team will using these lessons to guide our work distribution, and as our leader/logistical coordinator, I will try to hold us accountable to our deadlines.

Yatin: I really enjoyed this lab. It enabled me to explore different methods of

processing LiDAR data and innovating interesting new perception techniques. I was also pleasantly surprised by the effectiveness of the dual-mode PD controller, which was quickly implemented close to the start of this research period. We were so invested in our new ideas such as applying the Hough transform, progressive regression, downsampling, and scan optimization that we ended up spending more than 40 hours on this lab over the past week. Echoing my teammates above, being creative is great, but it should be balanced out with a healthy dose of "don't try to fix things that work well!"