Final Lab Report

Team 8

Xavier Bell
Nicolaniello Buono
Mary Foxen
Elise Harvey
Eli Scharf

6.4200

May 14, 2024

Contents

1 Introduction

(Xavier)
The final challenge is an accumulation of all the skills we learned in the previous labs. It is comprised of two parts. The first, Mario's Circuit, is a racetrack lane following a competition in which our racecar must drive around a full track autonomously as seen in Fig. 1. The second, Luigi's Mansion, is a city driving challenge where our racecar must navigate Stata's basement to collect coins while avoiding pedestrians, detecting stop signs and stop lights, and obeying traffic laws. In Fig. 2 our racecar follows traffic laws by staying to the right of the middle line.

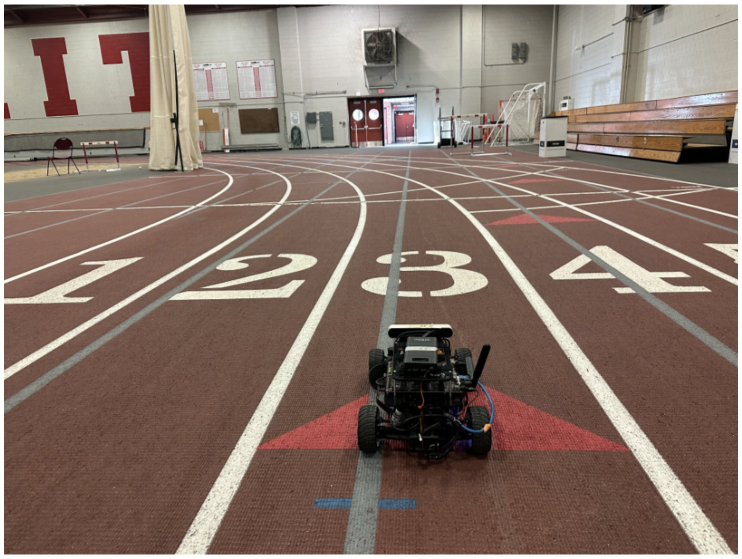

Figure 1: Overview of racecar maintaining its lane while traveling around the track. The racecar was forbidden from crossing into other lanes.

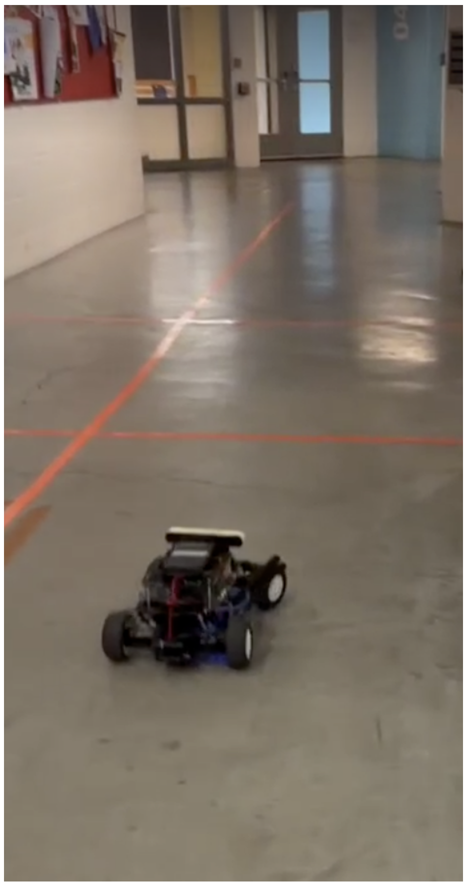

Figure 2: racecar car stays on the right side of the road as it travels to its destination. It used camera data to detect if stop signs and traffic lights came into the frame. It used its safety controller to stop if pedestrians crossed its path.

The two challenges represent different extremes of robot problem-solving. Mario's Circuit takes place on the controlled environment of a racetrack with a simple task to complete. The main problem is the careful refining and control optimization to race at high speeds. On the other hand, Luigi's Mansion presents a more chaotic and unpredictable setting akin to city streets, where the racecar must contend with a multitude of dynamic factors, including pedestrians and traffic signals.

These challenges are greatly relevant in the field of robotics and autonomous driving. Both challenges require image processing and detection. The detection and segmentation of lanes in Mario's Circuit and the detection of traffic lights and stop signs in Luigi's Mansion parallels a self-driving car's examination of its environment and movement through a chaotic city environment. Both challenges also require trajectory following which is highly important topic in robotics. Furthermore, Luigi's Mansion utilizes Monte Carlo Localization (MCL) from Lab 5.

2 Technical Approach

(Eli)
This lab involved two main challenges: Mario's Circuit and Luigi's Mansion. For Mario's Circuit, our autonomous racecar needed to drive as fast as possible around a standard 200-meter track. We had to ensure the racecar stayed in its starting lane. We also needed to ensure the racecar had a safety controller. The safety controller ensured the car stopped if another car entered our lane or our car veered off course. For Luigi's Mansion our car needed to drive fully autonomously through Stata's basement. The car received three destination points. The car needed to drive to these three points in order to abide by all traffic laws. The racecar needed to stay on the right side of the road. It was only allowed to cross to make a U-turn to reach a destination point. The racecar also needed to stop at all traffic lights and stop signs. We also needed to watch for pedestrians. Pedestrians could cross at any time. We implemented a safety controller that would stop the car if pedestrians crossed in front of it or any other obstacle moved in front of the racecar's path.

2.1 Controller

(Mary and Nico)
In both portions of the final challenge, we used a pure pursuit controller with Proportional Derivative (PD) control on the steering angle. The goal that was passed to the pure pursuit controller was chosen with different look-ahead parameters for each challenge. But, the core implementation largely remained the same for both challenges. When a goal is determined via look ahead parameters and other algorithmic constraints (such as choosing a goal point along a planned path at some look ahead distance away from the car), we command the car to move toward the goal using Ackermann steering geometry.
Our early implementations for pure pursuit control made the car oscillate causing it to incur penalties. This was particularly acute in the Johnson challenge as the car often disqualified itself in testing by driving away from its lane and/or away from the track entirely. Adding PD control on the steering angle eliminated significant oscillations and allowed the car to drive reliably in both of our challenges. The proportional component of the control solely depends on the error, which is our steering angle, multiplied by our `P` parameter.
The derivative component accounts for the rate of change of the error and helps to dampen oscillations. The derivative is calculated as the difference between the current error and the previous error, and a higher rate of change would lead the derivative component to take more control.

As an example, if we have a previous error/steering angle x, a goal that generates steering angle c via Ackermann steering geometry, and proportional and derivative constants `P` and `D`, respectively, the final steering angle we obtain with PD control is:

$$\text{Steering Angle} = \mathtt{P}c + \mathtt{D}(x - c) \tag{1}$$

2.2 Safety Controller

(Eli)
Our safety controller dynamically adjusted the stopping distance based on the racecar's current speed. The safety controller used Light Detection and Ranging (LiDAR) data to gauge the distance of obstacles ahead. The stopping distance was calculated based on the exponential as seen in Eq. 2. If the LiDAR scan detected an obstacle within the calculated stopping distance, the safety controller overrode the drive commands and stopped the car.

$$\text{Look Ahead} = 2 \cdot e^{(\text{velocity}-3.0)} + 0.3 \tag{2}$$

2.3 Mario Circuit

(Eli)

2.3.1 Line Segmentation

We considered different approaches to detect the track lines. The first approach used only color segmentation to detect the point and followed the point offset by some constant to keep the racecar in the center of the lane. We did not implement this approach because we feared the car would detect other lines on the track and would lose its path.

Our approach used color segmentation and Hough transforms to find the white lines on the map. We applied a mask to the initial image and filtered out all HSV colors that were not white, as seen in Fig 3.

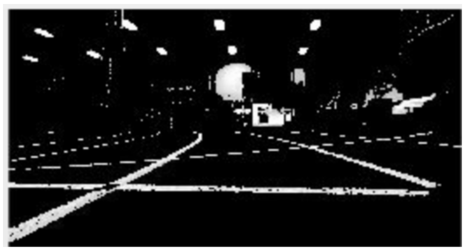

Figure 3: All colors except for white were filtered out. This allowed the racecar to clearly see the white lines of the track.

After all other colors were filtered out, we applied a Canny edge detector to the image. The Canny detector finds changes in intensity in the image. Anywhere that has a gradient or change is marked as an edge. All other colors in the image were represented as black pixels. This made differentiating white from black an easy task with little noise. A Hough transform was then applied to the image. Hough transforms represent each edge pixel in the parameter space as a sinusoidal curve. The intersection point of these curves represents a potential line in the original image. Each curve 'votes' for lines that pass through it. Peaks are calculated by summing the votes for a specific location. After processing all the points, peaks indicate the most likely lines. As seen in Fig 4, the Hough transform algorithm outputs all possible straight edges in the image. The Hough transformation algorithm is fairly robust to noise. It allows users to fine-tune parameters for finding lines in the image. The robustness and flexibility proved helpful for finding lines and fine-tuning the algorithm for better performance in our specific case.

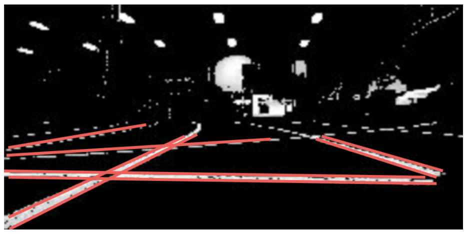

Figure 4: Hough transform finds all straight lines in the image. There is little noise and the lines align well with the actual track lanes. Tuning the parameters of the Hough transform allowed us to remove unwanted noise or lines that were too short.

With all the straight lines, we needed to decipher which lines the autonomous car should follow. Several lines cross the track in odd directions and create noise for the racecar. To refine our data, we filtered out lines that had a slope of less than one-fifth. Lines with slopes less than one-fifth ran perpendicular to the car are were likely not part of the track. Filtering these lines stopped our racecar from veering off the track. We also cropped our image to prevent lines from adjacent lanes from adding noise.

Sometimes our algorithm would return more than two straight lines. To choose the best line for the right and left sides of the lane, our algorithm iterated through all the lines and sorted them based on their starting points. The reference frame for the lines placed the point $(0, 0)$ in the top left corner of the image. Thus, the algorithm chose the line with the greatest starting y-value and smallest starting x-value as the corresponding left lane. Alternatively, the algorithm chose the line with the greatest starting y-value and largest starting x-value as the right line. After filtering out data, our racecar generally saw two parallel lines, as seen in Fig 5. We returned these lines to the controller.

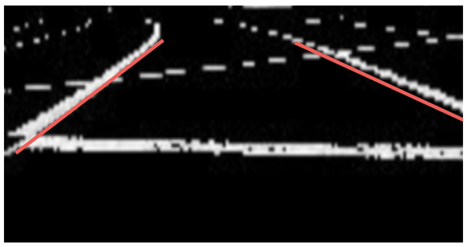

Figure 5: Final lanes found after filtering image. By removing horizontal lines, clipping the image, and finding the most likely starting point our algorithm returned two parallel lines corresponding to the lines on the track that we needed to follow.

2.3.2 Track Specific Control

With two lines detected, our racecar applied a homography transformation to orient the lines in the car's frame. The homography transformer converts between image pixels and real-world coordinates in the racecar's frame. In the car's reference frame motion forward corresponded the positive x direction. Motion to the left corresponded to the positive y direction.

We calculated the pursuit point once the lines were oriented in the car's frame. If the car saw two lines, our look ahead point was set as the point in between the two lines at a look ahead distance of 1.8 meters. If the racecar only saw the left or right lines, the algorithm returned a look-ahead point based on the slope of the line. For example, the slope became flatter as the lane became more perpendicular to the car. In this example, the car would need to add more offset in the y-direction to return to the middle of its lane. The algorithm found the line with a perpendicular slope to the single lane detected. The perpendicular line was converted to a unit vector that was multiplied by our constant offset (set as .45 meters). This determined the point on the y-axis that that racecar should pursue. The x-axis of offset was 1.8 meters. This is illustrated below in Fig. 6.

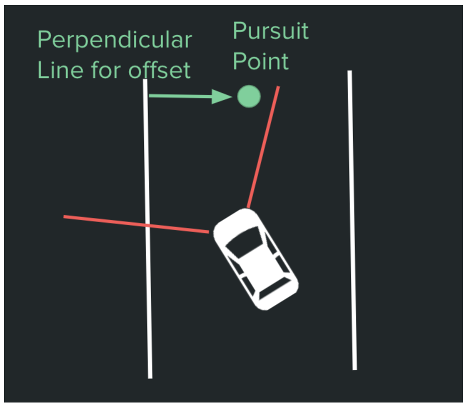

Figure 6: racecar only sees one line. Pursuit Point is calculated based on the perpendicular line to the slope of the lane seen. The y-offset is multiplied by the y-offset and the pursuit point is found

2.4 Luigi's Mansion

(Nico)
The objectives of Luigi's mansion include planning to collect three shellsTM at unknown locations–denoted by clicks on our map–and obeying traffic laws while navigating the Stata basement. To tackle this challenge, we separated the problem into five different deliverables: a state machine to delegate tasks, a driving component, traffic light detection, stop sign detection, and pedestrian detection.

2.4.1 State Machine

(Nico and Mary)
In the realm of autonomous vehicles, ensuring safe and compliant city driving demands a robust system capable of managing various tasks seamlessly. To address this challenge, a state machine was implemented as a fundamental component of our autonomous car project. This state machine provided high-level control for the vehicle, orchestrating its behavior in response to dynamic traffic scenarios including pedestrian crossings, traffic lights, and stop signs.

In order to follow ”city driving” traffic within our state machine, we included the different states ”drive”, ”stop”, ”departing”, ”park”, and FINISH. We enter the ”park” state when we arrive at a shell. We subsequently enter the ”departing” state for 3 seconds after being in the ”park” state for 5 seconds. We enter the ”stop” in the following cases: there is an obstacle directly in front of the car (which we assume to be a pedestrian); a stop sign is detected and we are not in the ’departing’ state; we are near a traffic light, see red, and are not in the ’departing’ state. Otherwise, we enter the ”drive” state. In this way, we are able to successfully handle each possible traffic scenario. See Fig. 7 for a visual representation of the states and transitions.

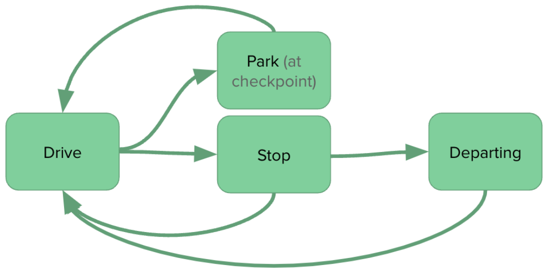

Figure 7: **A visual representation of the state machine that delegated tasks in the city driving challenge**. This allowed processes to run continuously. It told the racecar when to drive, stop, or plan a new path.

2.4.2 Driving Component

(Elise)
To drive around Stata's basement to clicked points in our map, we needed to find a way to drive from point A to point B while staying on the right side of the road. To do this, we combined three major components: (1) our path planning solution from lab 6, (2) the given trajectory of the orange line in the Stata basement, and (3) a pre-computed u-turn sequence.

First, we planned a path using Breadth-First Search (BFS) from the car's starting location to the closest point on the given orange line. This allowed us to locate the center line we should be following. Next, we used a subset of the orange line's trajectory that enabled us to follow the center orange line to where the goal point was. To ensure that we correctly obeyed traffic rules, we shifted the location of our car to the right by 0.3 meters when following the center line. Lastly, we included a u-turn sequence. If we were facing forward on the orange line and the foal point was behind the car, we would execute a u-turn at the start of the driving sequence. Additionally, if the goal point was on the left side of the line, we would execute a u-turn at the end of the driving sequence to ensure we parked on the goal point.

For the u-turns, we could not use BFS to plan the turn since it does not account for the feasibility of a planned path. In other words, the path returned by BFS would not account for the turn the racecar would need to make. With the assumption the u-turns would only need to be made when following the center orange line, we were able to pre-compute our own feasible sequence that the car could execute. So when a u-turn was needed, it would add the pre-computed sequence to the trajectory where needed. To get this trajectory, we iteratively tried different sequences until we got the tightest turn the car could make. The final points are shown Fig. 8. Note that to complete these u-turns, we assumed that there would be an area of 4 squared meters. This assumption was chosen by measuring the hallways given on the map and testing the sequence in different locations.

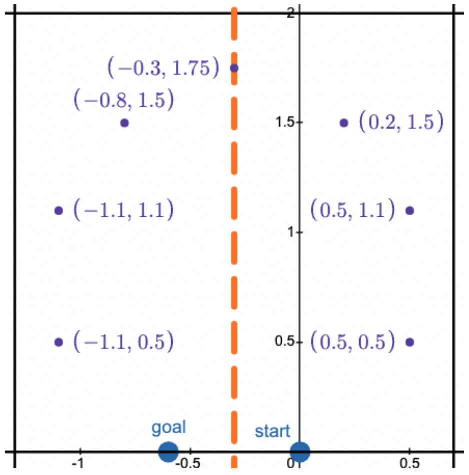

Figure 8: The pre-computed u-turn sequence relative to car's location at (0,0). These points were picked through experimental evaluation and testing.

Determining when we needed to execute a u-turn at the start of planning a path was simple. We simply had to track whether the previous trajectory followed a "forward" or "backward" path. If the next path was the opposite of the previous, we executed a u-turn before planning. Determining whether we needed to execute a u-turn at the end of the sequence was more difficult. To do this, we created a vector that points to the final point on the orange line we were following. This vector would point in the direction that the car would be going when it reached that location. Then, we created a vector to the goal point. If the dot product of these two vectors was less than zero, that means the goal point is left of the original vector–or in other words, it is to the left of the center orange line. If we found that a point was to the left of the line, we would add a u-turn at the end of the path to complete driving to to goal. See Fig. 9 for an example of a planned path with a u-turn sequence.

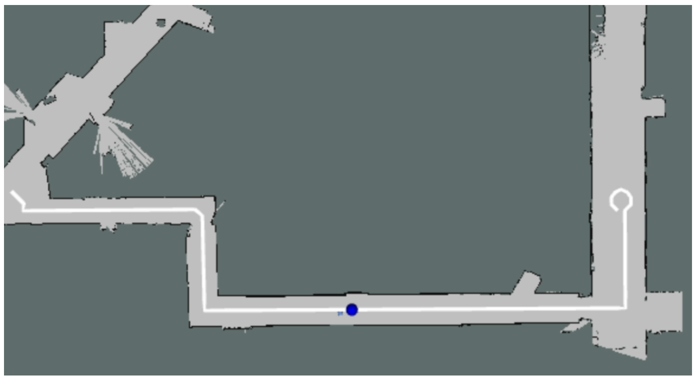

Figure 9: **An example of a planned path by our driving component.** As shown at the end of the trajectory, the car will execute the u-turn sequence. Additionally, note that the car is following the main section of the path shifted to the right to obey traffic laws.

2.4.3 Traffic Light Logic

(Elise)
For detecting traffic lights, we used a combination of tools. First, we used color segmentation to find if the light was set to green. We initially tried to recognize red but realized a red color segmentation would find the red bulletin boards, orange tape on the floor, and the wood of doors. Second, we used our localization algorithm (Monte Carlo Localization) to recognize when the racecar was in the vicinity of a traffic light. We were able to do this since traffic light locations were published before challenge day. This combination of localization and green recognition allowed us to tell the car to only drive past a traffic light if it saw a green light. To recognize the relative distance to a green light, we used the relative height in pixels of the green circle recognized to determine the relative distance. For example, we found that if the green area recognized was more than 10 pixels tall, then the traffic light was within a meter of the car.

If we were near a traffic light and the light was not green, the car was commanded to stop. Only once it recognized green could the car continue forward. In practice, there were several challenges to implementing this. First, if our localization was off, then the car would not recognize it needs to stop. This occurred on the challenge day since the starting area messed with our localization algorithm. Additionally, we had to crop the image that the car sees to make sure that it did not pick up on green from external factors like people's clothing. With this, the algorithm struggled to see the traffic light if the car stopped in the wrong location.

2.4.4 Stop Sign Logic

(Elise)
For the stop sign detection, we were given a machine learning algorithm that would recognize a stop sign in the camera's image. Unlike a traffic light, a stop sign would not "turn green" and allow the car to pass. To handle this, we created a "departing" state. Once the car recognized a stop sign, it would come to a complete stop. After coming to a stop, it would enter the departing state. This allowed the car to continue driving without being re-triggered by the stop sign in the camera. We set it so the departing state would only last for a set amount of time before reverting to the regular drive state.

Similar to our traffic light distance detection, we were able to measure the height of a stop sign in pixels at 1 meter and 0.5 meters away from the car to come up with a suitable distance range for the car to stop. We found this range to be 50 to 90 pixels. Due to time constraints, we were not able to integrate our working stop sign detector.

2.4.5 Pedestrian Logic

(Elise)
The pedestrian logic is very similar to our safety controller. Since we did not want to stop only at crosswalks–for example, if there was an unknown obstacle when driving–we created a generic checker that followed the logic of our safety controller. We used LiDAR data to check up to 0.5 meters in front of the car. To make sure that the traffic lights on the track did not trigger the stopping mechanism, we only checked approximately 10 inches to the left and right of the car. This ran regardless of location in case the localization algorithm was off and the car was headed toward other obstacles.

3 Experimental Evaluation

(Xavier)
Due to the nature of the challenges and the short time frame, we weren't able to conduct much quantitative experimental analysis. As our path planning algorithm will always return the most optimal path, finding errors was unnecessary. On race day, we encountered numerous errors that led our racecar astray. These would affect our ability to meaningfully categorize the performance of our system. While we were only able to complete a handful of trials, we have video footage of many of our attempts.

3.1 Mario's Circuit Performance

(Eli)
For the Johnson track trials, our team's best time around the 200-meter track

was 52 seconds with 0 infringements. Ideally, we should have been able to complete this loop in 50 seconds. We may not have finished the track in the ideal time due to small oscillations while the car was driving. Our racecar oscillated even with the PD controller applied. These oscillations added extra distance for our racecar. Additionally, in order to minimize the distance traveled, our racecar should have stayed as far to the left as possible. However, our pursuit point fluctuated, and hugging the left lane increased the risk of lane infractions if the pursuit point moved too far to the left. Thus, our team decided to maintain a pursuit point in the center of the lane.

Although driving at 4 meters per second had the highest risk of infractions, it led to our best overall time and score. As seen in Fig. 10 and Fig. 11, there is a trade-off between the fastest lap time and deductions based on infringements. As the car drives faster, it also begins to oscillate more. We chose to drive the car at 4 meters per second because we prioritized speed over accuracy of staying exactly in the middle of the lane.

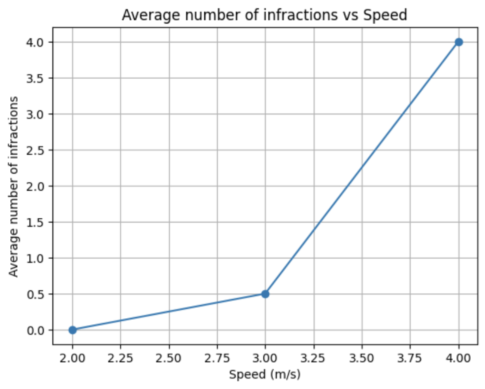

Figure 10: Average number of infractions based on the speed of the racecar. The is an exponential increase in the number of infractions as the car's speed increases. As the car travels faster, smaller steering command errors make a larger impact. Thus we saw an increase in infringements with greater speed.

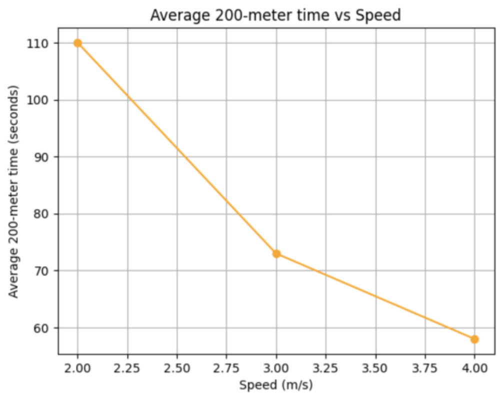

Figure 11: Average time to complete 200-meter lap based on speed. As the racecar's speed increased it completed the lap faster. The number of infractions and total time determined the speed at which we raced.

3.2 Luigi's Mansion Performance

3.2.1 Race Day

(Nico)
The car behaved erroneously on race day. Paths that worked when testing independently had the car making unexpected turns that deviated from the ground truth path. We believe that the new obstacles present on race day — the dozens of people and several laptops, racecars and routers — caused localization inaccuracies which made our car deviate from the ground truth paths. When our car needed a manual intervention, the physical lifting and moving of the car caused it to further lose localization, exacerbating path following inaccuracy.

3.2.2 Independent Testing

With fewer obstacles present, the path following worked flawlessly. Unfortunately, we were unable to test stop sign detection and handling due to time constraints and the stop signs being placed too high, outside of the camera's field of view.

4 Conclusion

(Mary)
In this final lab challenge, our team undertook a comprehensive application of racecar principles including algorithmic design and system integration to address the demanding tasks of Mario's Circuit and Luigi's Mansion. These challenges necessitated the deployment of sophisticated control systems, image processing

algorithms, and state-of-the-art navigation strategies to achieve autonomous operation in diverse and dynamic environments.

In the Johnson Track portion, our autonomous racecar struggled during some laps but performed remarkably well on other laps. We recorded the fastest lap at 52 seconds with zero infractions. To make the racecar more robust to lighting changes, we hope to implement more heuristics that help the car localize itself along the track and within its lane. In addition, we hope to better filter adjacent lanes in order to decrease added noise.

For Luigi's Mansion, our team performed well when testing. However, for the final challenge, our autonomous car did not perform as well as expected. In order to address the last-minute errors we encountered, our group will investigate our localization algorithm. Specifically, we hope to improve its accuracy even when people are standing near the LiDAR scanner. Red light detection and color segmentation can also be improved. By detecting perpendicular orange stopping lines, our car can determine when to look out for traffic lights and stop signs. Finally, further testing would allow us to determine a more accurate traffic light height in the frame so that we could find the best view of the traffic light for red light detection.

5 Lessons Learned

(Xavier)
In this lab, I continued to refine my CI skills in planning, communicating and presenting. Due to the longer time frame of this lab, we set deliverables earlier in the week to track our progress. While we ran into some difficulty debugging at the end (as we always do), we performed very well on the racetrack and better than most in city driving. I had 4 final projects due in 3 days and struggled to manage my time and priorities. I tried to communicate my absences to my team, but I could have certainly done better. I was more active in report and briefing creation than in previous labs which was I comment I made in the Lab 6 report.

With regard to the technical aspect, I spent the vast majority of my time on the racetrack. It was a deceptively difficult problem. The controls didn't take much time, but a lot of effort was made in consistently finding the point to target. Also dealing with a physical robot was very finicky. We had our entire code base corrupted on the robot multiple times. I wish I played a larger role in the city planning challenge as I think I could have learned a lot.

(Nico)
With regard to logistics and CI aspects, I'm happy that we began brainstorming early and happy with the briefing we gave. I think that the long time frame of this lab lent itself to a lack of urgency in producing technical deliverables earlier.

On this project more than the previous ones, we spent considerable time trying to integrate systems in the days leading up to the challenge. I also think I could have done a better job communicating my illness in the week leading up to the challenge — especially considering I was implementing a critical system of the city driving challenge — and the resulting catch-up I had on multiple final projects and assignments. I'm proud of my team for how they took on the work on the
For the technical part of this lab, I focused most of my effort on Luigi's Mansion challenge. I was working on creating a robust controller that would have the car iteratively follow a line that bisects the tape and the right wall, and then drive to a Shell when it is in "line of sight," which I determined by checking if. I also created an implementation for u-turning that used primitive model predictive control to inform whether a multi-point turn is necessary. Unfortunately, I was sick for most of the week leading into the final race day and did not finish and/or test any of my implementations. After communicating with the team and determining what would be best to focus on, once I was feeling better I worked on general debugging of the city driving, spending time getting the physical car working, segmenting color to detect the green light of traffic lights, and some minor tuning of PD control on steering angle.
(Mary)
In this final challenge, I am proud of our group for starting the brainstorming process early. By doing this we were able to determine important higher-level details before undertaking deeper, time-intensive tasks. However, because of the higher complexity of this lab and the scale of these challenges, we spent a lot of time debugging last minute errors that occurred in the real world. In the future, I hope to remember this lesson by testing more early on, even if the final solution is not complete in simulation.

On the technical side, I felt that this final challenge allowed me to more easily see how our lessons learned in past labs may be integrated to achieve a more complex task. It was a very rewarding experience to get the chance to bridge so many different topics together and deploy it on one final autonomous system. There was much last minute work to be done in the final hours of this project with debugging, especially with errors that could never have been predicted. In the future, I hope to test code earlier and often even before the system is fully complete.

(Elise)
For the technical part of the lab, I started my work on (1) creating and finalizing logic for a state machine, (2) integrating the given stop sign detector (that used machine learning) into the state machine, and (3) creating a red light detector for traffic lights. When one of our team members who was fronting the work on the driving portion of the city driving challenge got sick, I ended up also trying to create a solution as fast as possible. Because of this, this was the heaviest load I had in terms of deliverables in this class and I struggled since I was not prepared for it. We ended up testing our city driving very late in the timeline,

which had a hand in why we were unable to successfully integrate all the pieces. I spent hours iterating through ideas, coding, testing, and trying again. I also spent hours debugging the city driving solution and helping fine-tune the Johnson track solution.

For the CI aspect of this lab, I practiced shifting and planning out times to work on the lab. Since half of the team needed the racecar at the track and I needed the car in Stata, we had to schedule times to work on it. Additionally, both teams heavily relied on the camera which drains the battery, so we had to try and maintain the battery as much as possible. This was definitely a blocker in terms of getting our solutions to work in time for the final challenge.

(Eli)
For this lab, I spent the majority of my time working on Mario's circuit (Johnson track). The task was far more difficult than I expected. I expected that we could copy our code from previous color segmentation labs. However, in this lab the robot was traveling much faster than compared to previous labs. Johnson track required a great deal of fine-tuning different parameters and debugging the code. My debugging skills improved a great deal in this lab. My teammates and I spent time systematically working through different errors and functions. We wanted to ensure our code ran well and the algorithms we were using worked correctly.

For the CI component, I used many of the skills I have previously learned to write this report and make the presentation. Our team did not have as much time to write the report or make the presentation. In previous labs, I felt the need to constantly reference old papers and outlines in order to ensure I checked all the boxes. However, for this lab, I feel more confident in my ability to present well and write in a way that conveys all the information. I have seen myself and my teammates grow in written respect.