Final Challenge: High-Speed Lane Following and City Driving

Team 19

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

6.4200: RSS

May 15, 2023

1 Introduction - AY

Throughout the Robotics: Science and Systems course, we have toured a variety of robotics problems, including wall-following, parking, line-following, localization, and path-planning. For the most part, we have solved these problems with specific guidance from the course staff about what approach to use. In this final challenge, we apply techniques of our own choice to complete two open-ended challenges: track racing and city driving.

Although these two challenges seem quite different, we noticed that they can be somewhat reduced to the same problem – line following in the vicinity of distractors. In our work, we followed the same iterative approach for the vision portion of both problems. For the first iteration, we rely on color segmentation and bounding boxes to identify points on a line to pursue. For the second iteration, we explicitly detect lines of a certain color by utilizing the Hough transform in the image and use geometric information to compute pursuit points. We found that pure color segmentation proved useful for simplifying complex environments like the city, while explicit line detection excelled for shallower turns like on the track but only increased complexity when attempting to handle sharp, 90° turns. We will further examine the benefits and drawbacks of each approach in the sections to follow.

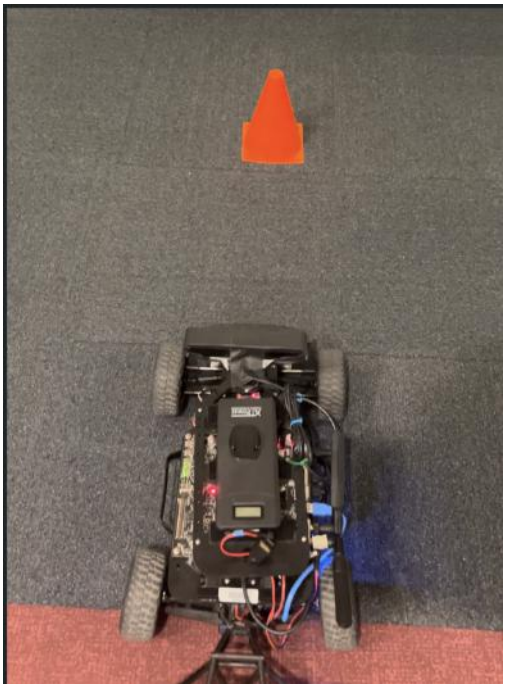
2 Part A: One Last Ride

2.1 Technical Approach

In this problem, we are randomly assigned a lane $l \in \{1, 2, 3, 4, 5, 6\}$ of a circular track, and our objective is to minimize the time it takes to traverse the entire track while staying within the white lines defining the boundary of l. The problem can be divided into two portions: vision and control.

2.1.1 Initial Vision Approach: Bidirectional Line Detection - AY & OM

As we learned from previous labs, we decided to start simple by adapting the cone detector and parking controller from the Visual Servoing lab. In that lab, the task was to park the car in front of an orange cone. One problem we had to solve on the way was segmenting and localizing the cone, which we later modified to create an orange line follower 1. The problem of staying between two white lanes was similar. We expected



(a) Cone Detector for Parking

(b) Line Detector Adapted From Cone Detector

Figure 1: As inspiration for the initial approach for the final race, we adapt the line following solution from Visual Servoing further in order to handle lane following. As shown in (b), we had already adapted a cone detector in (a) to perform line-following.

there to be only a few differences from the lab 4 code: the upper and lower HSV segmentation bounds should select for white rather than orange and the algorithm should detect one bounding box for each line rather than the one for the cone/line.
Hence, at a high level, our initial pipeline was the following:

1. Take in the camera image and remove irrelevant/confounding regions.
2. Split the remaining image into left and right halves.
3. For each half, perform color segmentation to draw bounding boxes around the respective white lane line for that side.
4. Take the average of the bottom center pixels for each bounding box as the image pursuit point, $p_i = (u, v)$.
5. Use a homography mapping H to find $p_c = H(p_i)$, the mapping of p_i to the car coordinate frame.
6. Send p_c to the steering angle controller which outputs a drive command for the car to execute.

To assist in removing distractors, the rightmost and leftmost fifths of the whole image were cropped out. Additionally, in order to exclusively consider the portions of the lines that best defined the desired trajectory of the car, the top half and bottom eighth of the whole image were cropped as well. After these modifications were applied to the original image, the remaining image was split into its right and left halves and fed into the color segmentation function.

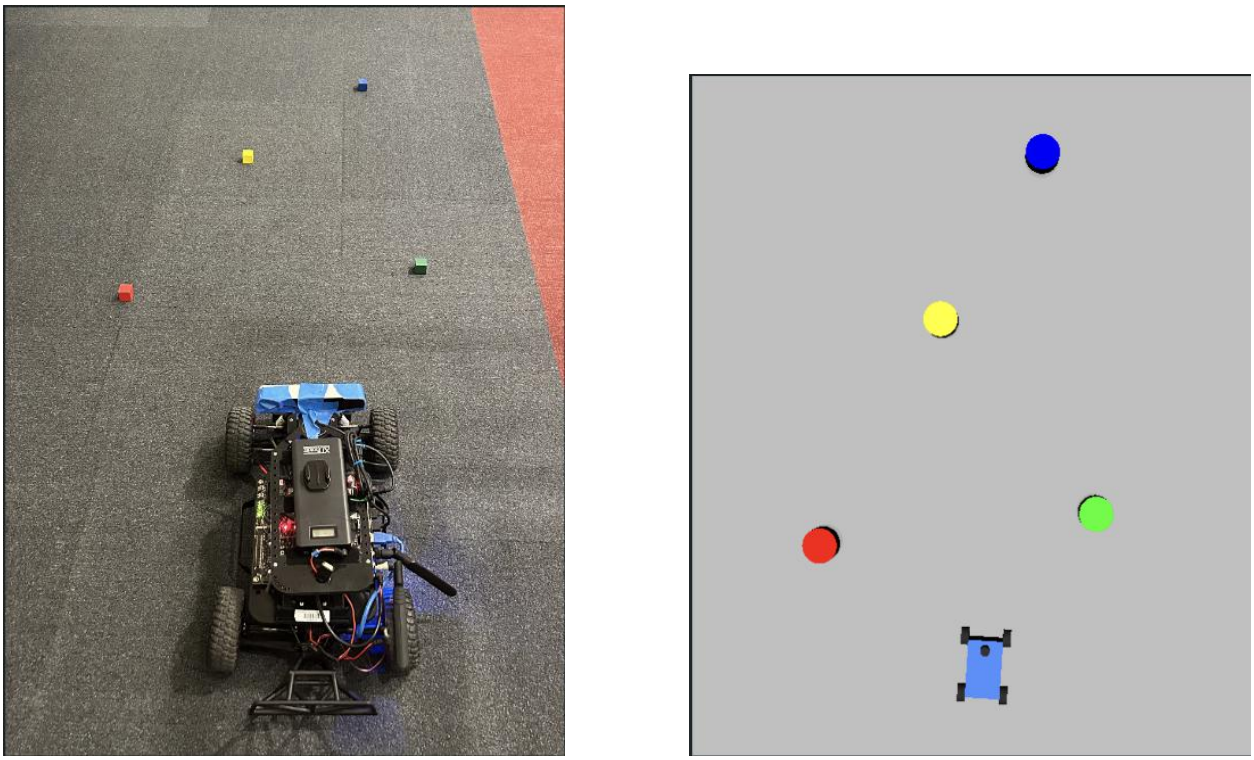
To perform color segmentation, we search through the cropped image and draw a bounding box to capture pixels that contain the color c which lie in the HSV wheel defined by lower bound (h_0, s_0, v_0) and (h_1, s_1, v_1). These bounds were determined through an iterative, qualitative process in which we applied the bounding

box detection to a pre-recorded ROSbag of the car driving around the track and observed the resulting bounding box. The final lower bound was fixed at $(h_0 = 180, s_0 = 180, v_0 = 180)$, and the final upper bound at $(h_1 = 255, s_1 = 255, v_1 = 255)$.

To find our homography mapping, we reuse the mapping H from the Visual Servoing lab. Recall that in homography, we find an invertible mapping $H : X \to Y$ where $X, Y \subset \mathbb{R}^2$ and X is the image space and Y is the frame defined by the car's reference. Although Y is naturally 3-dimensional, we can restrict it to a 2-dimensional subspace as we are only interested in points on the ground (i.e. for some fixed $z = z_0$). This constraint makes it possible to find such an H.

Finding H consists of collecting N samples $\{\mathbf{p}_i\}_{i=1}^N = \{(u_i, v_i)\}_{i=1}^N$ from the image space and the associated $\{\mathbf{w}_i\}_{i=1}^N = \{(x_i, y_i)\}_{i=1}^N$ measured from the frame of the car 2. We then minimize the following objective to find

$$H^* \in \arg\min_H \sum_{i=1}^{N} ||\mathbf{w}_i - H\mathbf{p}_i||_2^2. \tag{1}$$



(a) Real World Placement of Homography Samples

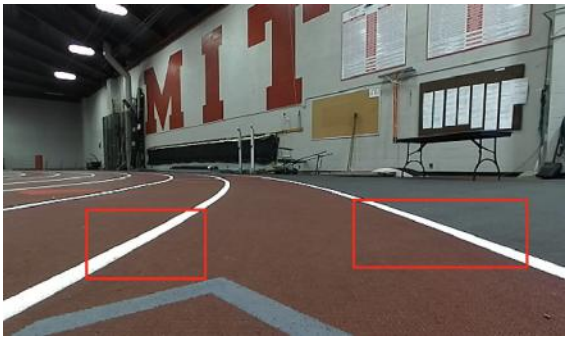
(b) Homography Samples Visualized in RViZ

Figure 2: To determine the homography mapping H, we sample N points that span the camera view. The figure illustrates a subset of the points that were used in calculating the homography. Subfigure (a) shows their place in the real world, while (b) shows the location determined through homography transformation of the car's camera image. As discussed in our briefing from the Visual Servoing lab, we attained an average error of 0.0274 (2 cm) for randomly sampled points outside of the training set.

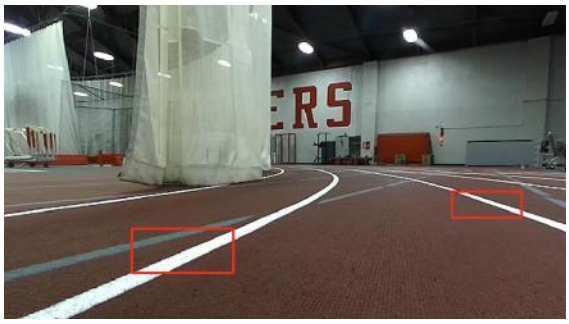
H^* can then be used to reliably map a point detected in the image space to a point relative to the pose of the car. This point is then sent to the controller described in Section 2.1.3.

Difficulties Accumulate Although simple and requiring little change from the Visual Servoing lab, the above approach suffered from drawbacks that exaggerated errors. The main difficulty was that there were

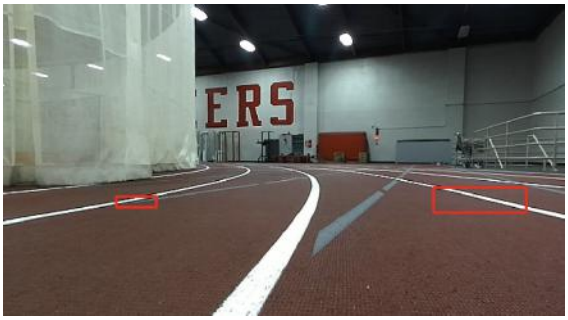
too many parameters to tune, and tuning to solve a specific case, like when the car is close to the edge of its lane, led to failures in other cases. Our segmentation would often lose track of the lanes or pick up distractors (e.g. horizontal lines, lines from other lanes) even when the image was cropped further. While we were able to achieve acceptable performance by incorporating logic that instructs the racecar to harshly steer back into the lane when no lines are detected, this hard-coded logic is bad practice and led to unpredictable behavior. Examples of various successes and failures of the color segmentation approach can be seen in Figure 3.



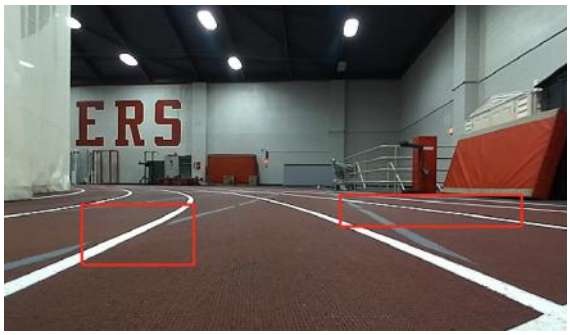
(a) Color segmentation success for straight-away



(b) Color segmentation success for sharp turn



(c) Color segmentation fail for uncentered car



(d) Color segmentation fail by detecting outside line

Figure 3: Two examples (a, b) of the color segmentation approach succeeding in detecting the correct lines, and two more examples (c, d) of the approach failing. The segmentation frequently fails when the car is not centered in its lane (c), as the image cropping can not be done dynamically, and therefore, the correct line is cropped out. Furthermore, it can catch outside lane lines (d), skewing the bounding box of one of the lines and giving an undesirable steering angle.

We noticed that the main factor distinguishing our desired lines and distractors was the lines' slopes. Color segmentation on its own is not able to account for this issue, so we turned to Hough lines to derive a more robust controller.

2.1.2 Minimizing Tunable Parameters by Transitioning to the Hough Transform - OM

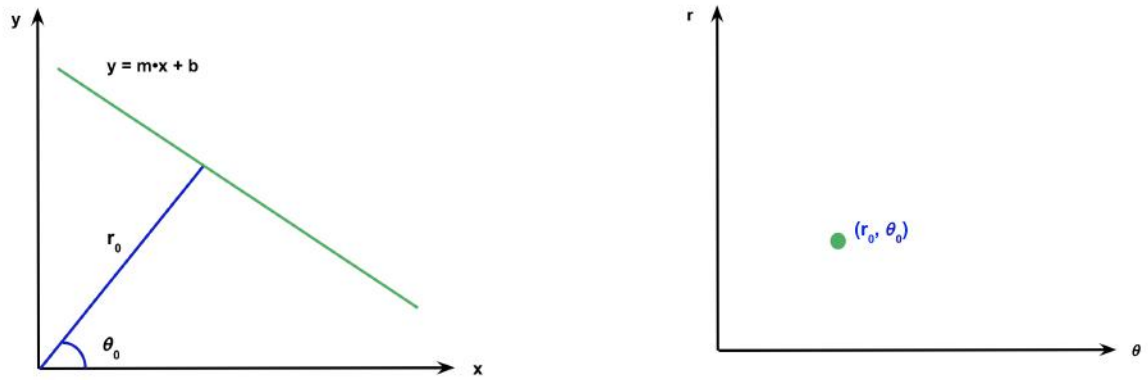
Having spent a significant amount of time tuning the color segmentation implementation with limited success, the team decided to consider more reliable methods for line detection. We landed on the feature extraction method known as the Hough Transform. For the purpose of this lab, we used the most basic form of the Hough Transform, which is to identify lines in the image, although the Hough Transform can also be adapted to identify more complex shapes. In simple terms, the Hough Transform is cleverly able to transform points to lines and lines to points by transforming the coordinate system to the "Hough Space". This tool can be very useful for detecting straight lines in an image.

The procedure for the Hough line detection implementation is as follows:

1. Take in the camera image and remove irrelevant regions.

2. Filter out all pixels outside of the same color range as stated in 2.1.1.
3. Utilize Canny Edge Detection to further reduce the noise of the image.
4. Perform Hough Transform on the left and right half of this modified image to identify all lines.
5. Filter lines based on slope and vertical intercept to find the desired lane lines.
6. Determine the average horizontal value between the lines at a set vertical lookahead distance l_d to find p_i.
7. Use H to find p_c in the racecar's coordinate frame on the ground.
8. Send p_c to the steering angle controller.

In order to understand the Hough Transform on a deeper level, we must start by defining the convention used for transforming the coordinate system as described earlier. A line in cartesian space is traditionally classified using its slope m and y-intercept b, yielding the equation $y = m \cdot x + b$. Thus, a line can be represented as a point (m, b) on a new set of axes. However, in image processing, this representation brings forth errors with representing vertical lines, for which m is undefined. A different convention, which utilizes the polar space, does not run into this issue; as opposed to a line being defined by m and b, the Hough Transform defines a line by its orthogonal distance to the origin r and the angle θ from the x-axis to this orthogonal line. Now, a straight line with any slope can be represented as a point (r, θ) on the respective axes, shown in figure 4. This coordinate system is known as the Hough space.



(a) The standard representation of a line in cartesian space

(b) The representation of a line as a point after being transformed to Hough space.

Figure 4: An example case of a line being represented as a point in Hough Space defined by its perpendicular distance from the origin, labelled r_0, and the angle from said distance to the x-axis, labelled θ_0.

A point in cartesian space can be considered the intersection of an infinite number of straight lines. If we transform all of these lines into points in Hough space, the resulting curve produced resembles a sinusoid. So, in totality, the Hough space transforms a line to point and a point to a sinusoid.

Now that we can transform both points and lines in Hough space, we would like understand how this transform can be applied for line detection in an image. Say a set of points (pixels) on a given line in an image are transformed to their respective sinusoids in Hough space. The particular (r, θ) that these sinusoids intersect at define the line that the points were sampled from in the original image! This phenomenon can be seen in figure 5

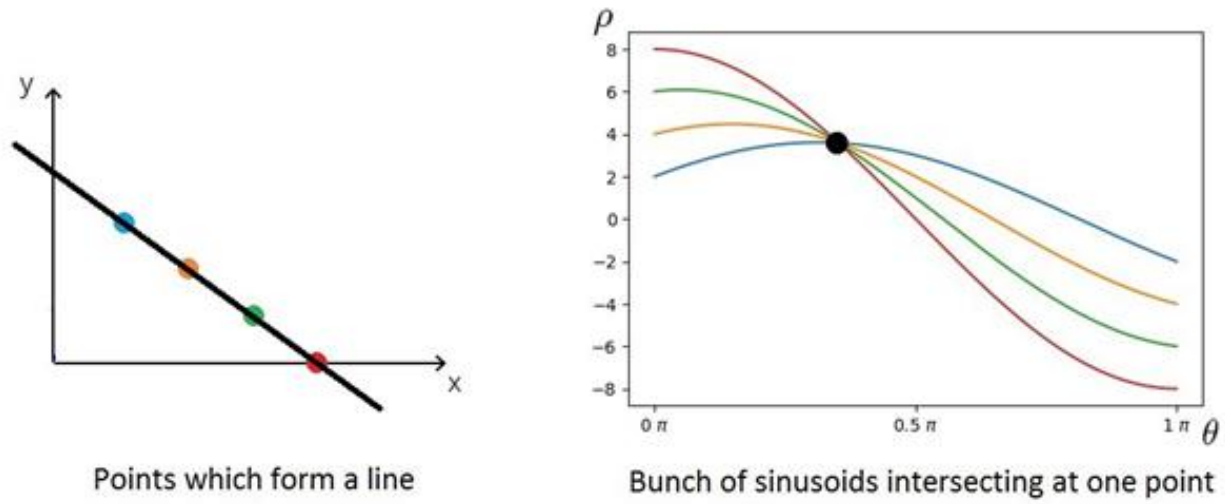


Figure 5: A visualization of points sampled on a straight line in cartesian space (left) and their associated sinusoids once transformed to Hough space (right). The sinusoids intersect at a single point, and this point defines the specific r and θ values that represent the original straight line. This image uses ρ convention for perpendicular distance, which is equivalent to the r used throughout this section. D. Biswas, "Hough Transformation," Medium, Jun. 28, 2021. https://dibyendu-biswas.medium.com/hough-transformation-1cec6541241b.

Integrating the OpenCV Hough Transform implementation on the above-described modified image yields a list of lines in the image in their Hough space representation. In most cases, the line on the left side of the image with the greatest b value is the left lane line, and the line of the right side with the smallest b value is the right lane line. The b can be found using r and θ as seen in equation 2.

$$b = \frac{r}{\sin\theta} \tag{2}$$

Due to this distinction, we found it useful to perform Hough line detection on the left and right halves of the image separately. However, this logic only succeeds if the only white lines in the image are those of the seven lane lines of the track. However, if there are horizontal lines crossing through the track, such as those found on the MIT Johnson Track, these lines will trump the lane lines in the max/min b criteria. Therefore, further filtering based on the θ value of the lines was necessary in order to handle noise. To filter out the correct lines with near-horizontal slopes, lines with $\theta \in [\pi \cdot (1/2 - 1/8), \pi \cdot (1/2 + 1/8)]$ on the left half of the image and $\theta \in [\pi \cdot (1/2 - 1/13), \pi \cdot (1/2 + 1/13)]$ on the right half of the image were ignored. The different ranges for the different sides of the image are a result of the left perspective bias of the camera, as the left lens of the two lenses is used for image processing.

With the two lane lines confidently identified, the horizontal value of the two lines is averaged at a lookahead distance of 120 pixels from the bottom of the image, or approximately 1.5 meters from the car's baselink coordinate frame (determined using H). This pixel, p_i is transformed to p_c using H, and finally sent to the steering angle controller, which aligns the car with the point while publishing a pre-set constant velocity magnitude.

2.1.3 Steering Angle Control Using Convex Combination of PP and PID Outputs – OM

Our line-following controller from the Visual Servoing lab utilized solely PID control to minimize the angle between the car's orientation and p_c. However, we found that this solution failed for speeds over ~ 3 m/s. We believe that this issue is due to PID acting quickly to reduce the error angle to the setpoint, which at high

speeds can lead to wild overcorrections. As we intended the car to run at its maximum speed of 4 m/s for the final race, this behavior was not sufficient. Pure pursuit (PP) is able to achieve a much more stable path to a point on a trajectory, but has limited reactivity to changes in curvature due to the constant steering angle nature of the path produced. Such limited reactivity can lead to oscillations around the setpoint, which we indeed observed when attempting to use a full PP output.

These drawbacks led us to the idea of using PID while adding a PP stabilizer to the final control signal. A convex combination of PID and PP control allows for the reactivity of PID and the stability of PP to complement each other, while reducing the drawbacks of both. The PID output was scaled by a weight γ, and the PP output was therefore scaled by $1 - \gamma$. Through qualitative observation, it was determined that the PID portion of the control signal should be weighted slightly more than that of PP in order to maintain a level of reactivity necessary to handle the changes in curvature of the track. The final weighting was determined to be $\gamma = 0.6$.

2.2 Safety - OM

With reliable lane detection and following, it is unexpected that the car will leave its assigned lane; however, we were aware that it may be necessary to push the algorithm to its limits in order to reach competitive speeds for the final race. Therefore, it is important to handle this case in order to reduce collisions and ensure the safety of the car when racing adjacent to others. We determined that the best course of action was simply to stop the car if it exited its assigned lane, as we believed it important to prioritize the safety of our/other teams' cars over the time required to complete the lap. Risking ruining the car's hardware is a much worse outcome than simply having to take the time to manually place the car back in its lane and gaining a few seconds on the race time.

In terms of knowing when the car has left its lane, it is likely impossible to distinguish one lane from another based solely on the car's camera image. However, it *is* possible to detect when a lane line is being crossed over, which indicates the car is leaving its lane. In order to find a defining characteristic of this transition between lanes, we manually drove the car across the lane line and observed the behavior of the vision element of the algorithm. It was observed that the car consistently lost sight of the line on one or both halves of the image.

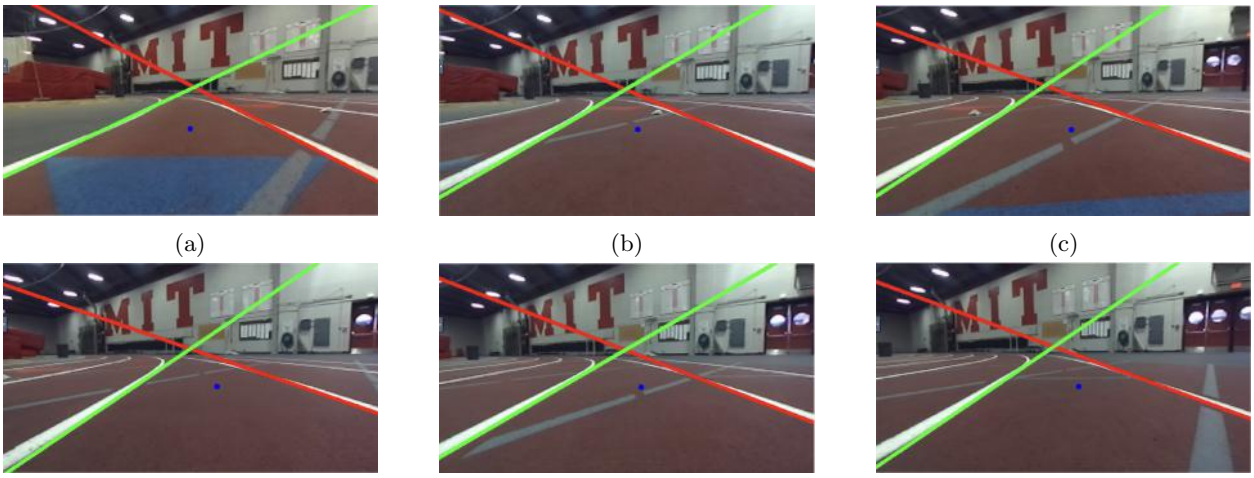
We therefore designed a very simple state machine in order to handle this transition event and react accordingly. The state machine checks if the car has lost sight of one or more lines, and if it has, publishes an immediate complete stop command to the car. The car will then only continue driving once it has been reset to see both lane lines. This feature was tested on speeds up to 4.0 m/s, in which it worked reliably. Luckily, on the actual final race day, this feature was not necessary, as the car never left its lane.

2.3 Evaluation

We now move toward evaluating the performance of our final approach. We verify that it comfortably completes the race regardless of the assigned lane number and tune our proportional gain to reduce oscillation.

2.3.1 Qualitative Results: Reliability of Line Detector - AY

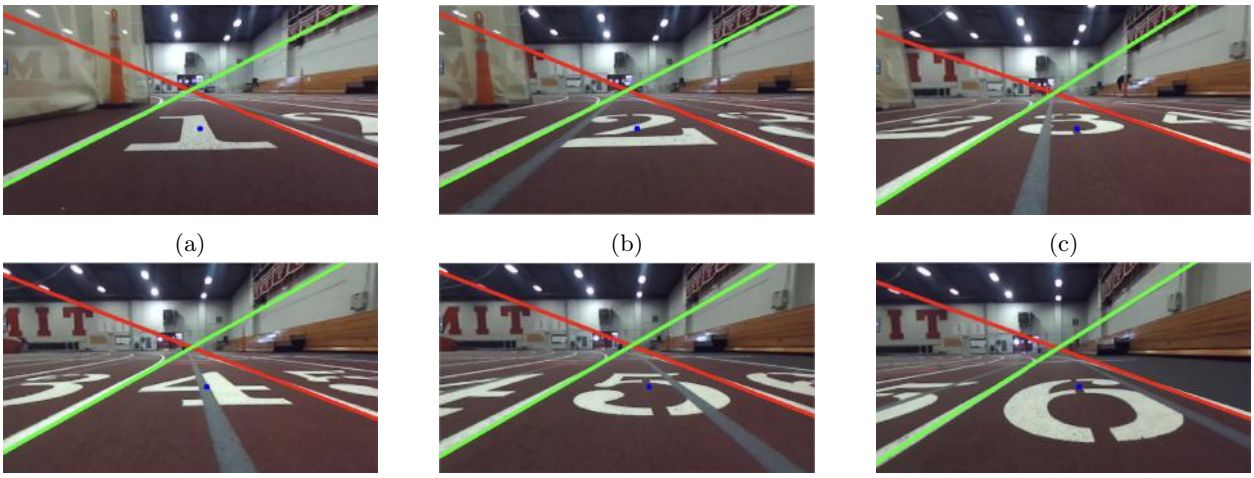
We qualitatively evaluate the reliability of our line detector by placing the car in a variety of positions that are representative of the cases it will encounter around the track. We visualize the lines representing the lane boundaries along with the derived pursuit point. As mentioned in 2.1.2, we decided to use slope-filtering thresholds that depended on whether we were looking for the left or right line due to both the camera perspective bias and because we were going counterclockwise around the track. Additionally, the color segmentation needed to be more conservative due to lighting differences in different regions around the track. After tuning the color segmentation bounds, image clipping amounts, and filtering thresholds, we see that our detector is able to handle varying starting lanes, number distractors, horizontal distractors, and the straightaway cases (Figures 6, 7, 8). This evidence suggests that our line detector is able to function as desired in varying lanes and regions of the track.



(a) (b) (c)

(d) (e) (f)

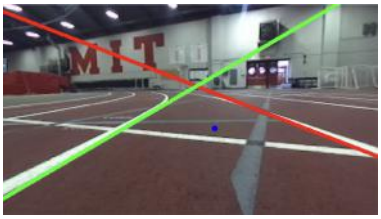
Figure 6: Visualization of the qualitative performance of our lane detector using Hough transforms as we vary the lane number with (a)-(f) corresponding to (1)-(6). The green and red lines identify the left and right lanes, respectively. The blue dot is our pursuit point; in all cases, it lies in right in the center of the lane despite the left-camera bias. We are able to successfully identify the lane boundaries despite the variance in curvature and potentially distracting grey lines.



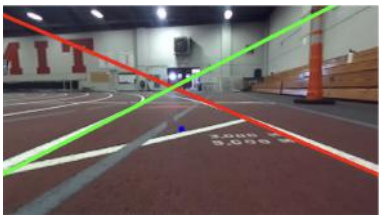
(a) (b) (c)

(d) (e) (f)

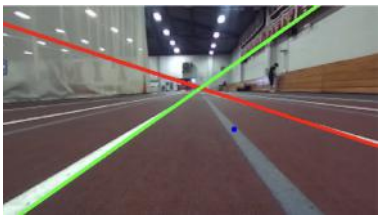
Figure 7: By masking out the center of the image and using Hough transforms to filter out distracting lines, we are unaffected by the lane numbers that appear at the end of the track. As in Figure 6, the green and red lanes identify boundaries and the blue point shows our pursuit point. Again, the pursuit point is centered within the lane as desired.



(a) Distractor for the right lane.



(b) Distractor for the left lane.



(c) Straightaway portion of track.

Figure 8: In (a) and (b), we provide evidence showing that our Hough line filtering allows us to remain unaffected by distractors that resemble lane lines even when the lines continue past the center of the image. In (c), we examine the simple case of a straightaway and confirm that straight, gray lines are not affecting the line detection.

2.3.2 Quantitative Results: Speed and Lane-Following Accuracy - AY

Evaluation of the Racecar's Speed To evaluate our car's racing performance, we ran tests going around the track at the max speed of 4.0 m/s. We count the number of lane breaches and how long it takes to finish the lap. As shown in Table 1, regardless of our starting lane, we have zero lane breaches and our final time hovers around 47.6 seconds. This average falls comfortably below a benchmark time of 50s. The consistency of our results reinforces the claim from 2.3.1 that our line detector is robust to varying curvature and that our controller is able to adapt as well.

Table 1: Time to Traverse Track for Each Lane

Lane Number	1	2	3	4	5	6	**Average**
Lane Breaches	0	0	0	0	0	0	**0**
Time (s)	47.58	47.55	47.75	47.65	47.62	47.69	**47.64**

Evaluation of The PID+PP Control Algorithm In order to evaluate and tune the pure pursuit and PID, we decided that the largest knob we could turn out of the three PID gains was the proportional gain. Empirically, the performance was very sensitive to perturbations in K_i and K_d, so we left them unchanged. Since K_p effectively dictates the reactiveness of our controller, we decided that it would be efficient to only tune one of K_p or λ. We fixed the control combination at $\lambda = 0.6$, so that the steering angle is given by $\delta = \lambda\delta_{PID} + (1-\lambda)\delta_{pursuit}$. We measured the angle deviation from the pursuit point at varying values of K_p as we traverse the first turn of the track, as shown in Figure 9. Assuming a standard track, the banking angle is approximately $\phi \approx 10^\circ \approx 0.17$ rad, so an accurate controller should have values no larger than ϕ. As suggested by Table 2, the value of $K_p = 0.2$ gives us the best performance, with an average angle error of $0.064 \approx 3^\circ$. Such behavior is quite desirable especially when considering that the turning angle around the circular portion of the track is roughly 10°.

Table 2: Average Angle Error From Varying Proportional Gains

$\mathbf{K_p}$	0.050	0.075	0.100	0.125	0.150	0.175	**0.200**	0.250	0.300	0.375
Average Offset (rad)	0.099	0.099	0.086	0.087	0.079	0.086	**0.064**	0.065	0.066	0.070

2.3.3 Quantitative Results: Lane-Following is Still Successful Even at Extreme Speeds - RS

Throughout this process, we had designed our controller to work when driving at 4.0 meters per second. However, on race day, we saw other teams racing their cars at speeds far beyond 4 meters per second, and our car couldn't match them at said speed.

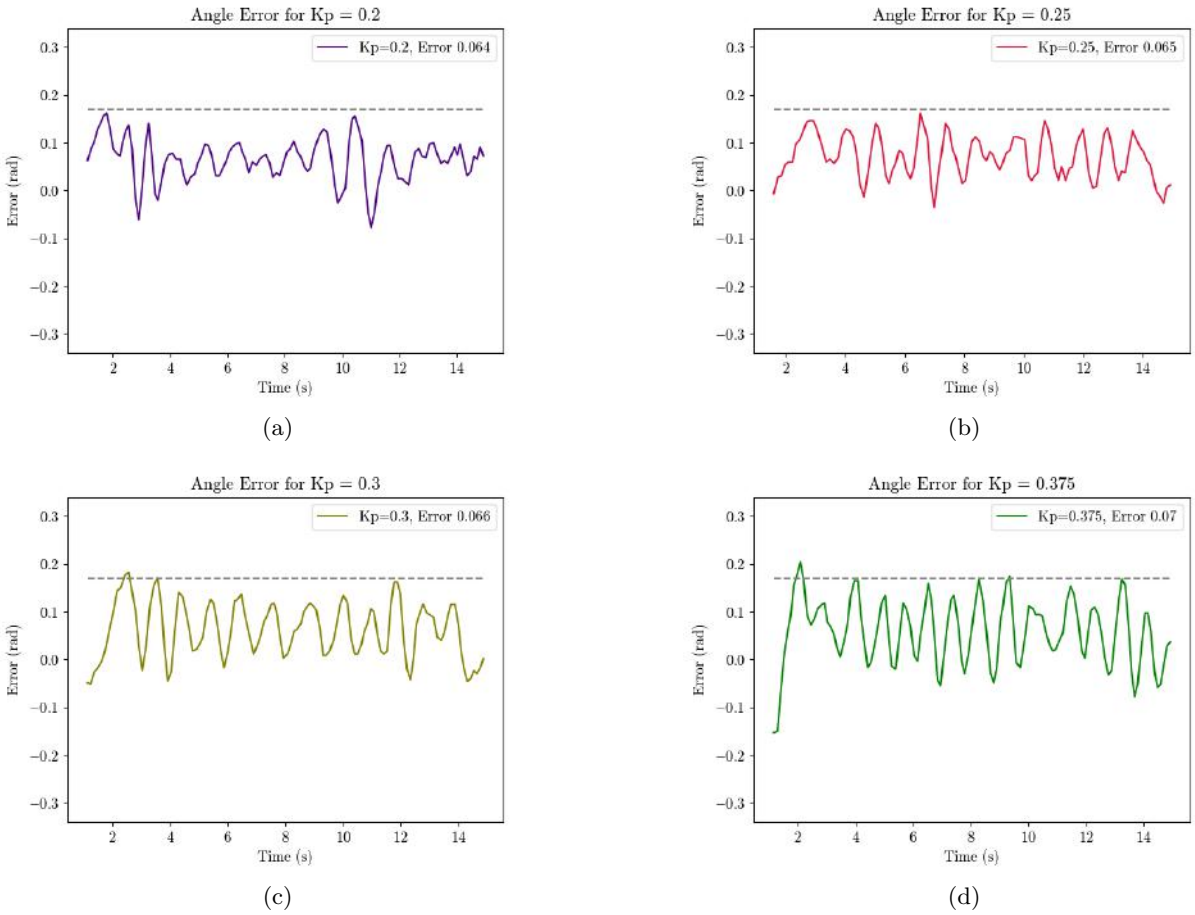


Figure 9: Visualization of the error over time as we traverse one of the circular portions of the track at varying K_p values. For conciseness, we show the plots for the K_p values corresponding to the four lowest average errors. The gray dashed line represents the natural banking angle of the track of 10°.. In plot (a), which illustrates the K_p value that was finally decided on, the fact that the car is reaching this line but not breaching it is very desirable and shows a lack of oversteering.

We wanted to see if our controller would work at higher speeds. We increased the allowed engine revolutions per minute (ERPM) and toggled our speed to higher values, and we noticed that our controller still worked remarkably well. Even as we increased it to 7.0 meters per second, our racecar still had no lane breaches as it raced around the track. It *would* oscillate much more within the lane, which is understandable given the speeds at which it is driving.

This result stands as a testament to our design choices, for even at higher speeds, which we had not tested nor imagined we would test our controller on, our controller still managed to circle the track without any lane breaches.

Specifically, it shows the good forethought in making K_p, our proportional constant, inversely dependent on speed. K_p is a scale factor that represents how aggressively our system is trying to correct for error. With a larger K_p value, our system will have a greater angle change responses whenever we deviate from our desired location, which is exactly halfway in-between the lane lines. However, this angle change, θ makes for a distance change of

$$speed \cdot \theta$$

in one second. Thus, we chose to have our K_p inversely-dependent on speed, for that way, our error re-

sponses wouldn't get magnified at higher speeds. As we could see from the tests we ran at higher speeds post-challenge, this idea paid dividends, for we didn't have to alter any parameters for it to work at higher speeds.

3 Part B: City Driving

3.1 Technical Approach

In this problem, we are trying traverse a city-like environment from a starting position to one of three portals representing destination points while avoiding obstacles (i.e. cardboard boxes). We chose the approach of utilizing the orange road line on the city floor to line-follow to Portal 3. Since the handling of stop signs was made optional, we split our following discussion into (1) exclusive line following and (2) our attempt to pause at stop signs.

3.1.1 Carrying Over Hough Lines from Part A - OM

In light of the success of Hough Lines for the vision approach in Part A, we set out with the intent of again using the Hough Transform to detect the orange line on the city floor. Detection was fairly easy, as we followed a very similar approach as in Part A:

1. Subscribe to camera image and remove irrelevant regions.
2. Filter out pixels outside of orange color range.
3. Utilize Canny Edge Detection to leave only the edges of the remaining image.
4. Perform Hough Transform on whole image to identify the road line.
5. Determine whether the road line is straight or if there is an upcoming turn, and find the pursuit point p_i accordingly.
6. Use same homography mapping H from part A to find p_c in the ground projection space.
7. Send p_c to steering angle controller.

Difficulty arose with item 5 above. Despite the fact that the lines in the image were identified, we needed a way to determine whether the lines represented a turn in the road line or not. We landed on the idea that for a straight-away, the two lines defining the sides of the road line tape will be practically parallel, but for a turn, there will be an intersection between the horizontal and vertical lines in the image. Occasionally, there would be intersections between the straight-away lines, but we handled this case by ignoring intersections if the slopes of the lines were too similar to each other. This logic allowed the algorithm to successfully distinguish a turn from a straight-away.

Now that the cases are distinguishable, we need to be able to find p_i in either case. For the straight-away case, the values of the lines representing either side of the tape were averaged at $l_d = 80px$ from the bottom of the image. For turns, we drew upon an idea from the previous Path Planning/Following lab, in which we drew a lookahead circle centered at the bottom middle pixel of the image, and found the intersection point of this circle with the horizontal line by reusing the circle intersect math from said previous lab. Seen in figure 10 are the two different cases, and the associated lines drawn.

Using a Hough Transform approach to this problem alleviates some of the weight of the line-detection problem from pure color segmentation, which can be unreliable due to lighting changes and similar colors in the surroundings, such as orange spots on the boxes.

Unfortunately, issues arose when the car began to commit to a turn. Early on in the turn, the pre-turn vertical lines would no longer be visible. Due to this issue, there would no longer be any intersections in

(a) Straight-away case for city driving hough implementation

(b) Turning case for city driving hough implementation

Figure 10: Two images illustrating the identification of the pursuit point for the straight-away case (a) and turn case (b). The lines identified in the image through the Hough Transform are shown in blue, and the lookahead radius in (b) can be seen in blue as well. In (a), the pursuit point is shown by the thick blue circle, and in (b), it is shown as the green dot.

the image, and thus the logic for distinguishing a turn from a straight-away would fail. We are sure that with some more time, we would have been able to create a very effective city navigator utilizing the Hough Transform, but within the time constraint of the final challenge, we chose to abandon this approach for the time being. We sought a simpler solution to the line detection problem through color segmentation.

3.1.2 Keeping it Simple: Back to Color Segmentation Approach - AY

While we hoped that the Hough lines would help the line detection algorithm be more robust to issues with the tuning of color segmentation, the approach was overly complex and ran into issues of its own. To keep the problem simple, we went back to the color segmentation approach. As stop signs were made optional, we began by framing the task as line following. We were able to modify our line following code from the Visual Servoing lab to follow the new shade of orange tape. Overall, the pipeline was very similar to that of the initial approach in Part A:

1. Take in the camera image and remove irrelevant regions.
2. Use color segmentation to draw bounding boxes around the orange road line.
3. Take the bottom center pixel of the bounding box as p_i.
4. Use H to find p_c in the ground projection space.
5. Send p_c to the steering angle controller. If no line is detected, then the car will continue driving at its last known steering angle.

Tuning Segmentation with HSV We soon realized the task was not as simple as it initially seemed. Due to the poor lighting of the basement, the range of colors for the line varied significantly at different points on the path. At some points, the orange line would become a shade of brown, and our initial color segmentation was unable to draw this fine line between our orange path and the brown boxes, causing it to sometimes make unpredictable turns into boxes (Figure 11). To address this difficulty with thin color boundaries, we carefully studied how HSV (hue, saturation, value) values were constructed. In particular, we noticed that the saturation value was crucial to get correct; the defining characteristic between the brown in the boxes and the brownish-orange on darker regions of the line was the saturation value, which often differed by two times. After tuning the saturation bound further, we were able to get much more reliable color segmentation. The final color bounds for the orange line were from $(h_0 = 15, s_0 = 125, v_0 = 97)$ to $(h_1 = 40, s_1 = 255, v_1 = 255)$ using OpenCV HSV conventions. During our debugging process, it was difficult to determine if incorrect behaviors (e.g. turning toward a box) were due to issues with the color segmentation or the controller. To determine which portion was causing the trouble, we attached a flashlight

to our car to correct for lighting issues and shift pixels more toward lighter shades of orange (see Figure 12). The flashlight allowed us to single out the performance of the controller, and tune that accordingly without facing segmentation issues. Some results of our color segmentation are discussed and shown in section 3.2.1.

Figure 11: When our color bounds are too loose, our car often ends up mistakenly detecting the bottom of the boxes as the line and skewing the pursuit point in the direction of the boxes. As seen above, the color of the box closely resembles the color of the orange line. It is almost certain that the box color will fall in the orange color bounds if the segmentation is done with insufficient care.

Modifications to Improve the Lab 4 Controller To reduce the prevalence of distractors in the car camera image, we make modifications similar to those from Part A in a preprocessing step. In particular, we mask out pixels above a certain height in the image and also slightly trim out the bottom. Ignoring this portion of the image allows us to focus on the region immediately in front of the car, while also ignoring objects that are way too close. In addition, we experimentally tune our PID gains as discussed in Section 3.2.2 to better handle the city environment.

3.1.3 Correcting Overturning Errors - RS

As shown in Figure 8, our racecar would sometimes collide with the first box along the city road. Sometimes, this error was because it incorrectly perceived the brown as orange due to lighting issues and the two shades being fairly close to one another.

However, this undesirable behavior also occurred when the car overturned and saw a strip of a future orange road line ahead. Our car follows a motor command which dictates it heads towards said location. Our car wouldn't understand that such a path goes through an obstacle. Therefore, it would mess up and mistakenly try to drive through the box or fail once our safety detector kicked in. Thus, we needed a way to prevent oversteering.

(a) Flashlight from the side

(b) Flashlight from above

Figure 12: In order to differentiate between issues between color segmentation and our controller, we mount a flashlight on the side of the car to brighten the shade of orange belonging to the tape. The flashlight helps the color segmentation differentiate between the tape and the box easier. Although there may be still false positives, picking the bounding box with the maximum size allows us to identify the line rather than the box regions.

As a means of solving this problem , we increased our derivative control constant, K_d. Derivative control seeks to slow down the rate at which we change our motor command, whether it be angle or speed. If we slowed down the rate at which we changed our angle, then we would not overturn and see the future road strip which caused this issue. Thus, we increased K_d, and upon increasing it, we no longer had this problem.

3.1.4 Incorporating Stop Sign Detection - OM

Due to the lack of distracting red objects in the city, we were able to again utilize color segmentation for the detection of the stop sign. Once a stop sign is detected, a state machine processes the information and determines the racecar's course of action.

One worry was that the color of the stop sign would be similar to that of the orange line, and therefore, the road line would be confused for a stop sign. This worry was handled by ensuring that the orange color bounds stated in section 3.1.2 were completely exclusive from those chosen for the red stop sign. This was done by examining the hue values necessary to properly segment for each color. With the lower hue bound for orange being set at 15 and the knowledge that lowering hue values closer to 0 increases the redness of the resulting color, we set the *upper* hue bound of the red segmentation to 14. The other values were then tuned empirically through a similar process as the orange and white values described earlier, resulting in a lower bound of $(h_0 = 6, s_0 = 80, v_0 = 120)$ and upper bound of $(h_1 = 15, s_1 = 255, v_1 = 255)$.

The color segmentation only detects the red octagon at the top of the stop sign, which is a distance off of the ground. As the homography mapping H only works for converting pixels to points on the floor, this floating bounding box is not desirable. In order to find the bottom of the stop sign pole, the proportions of the sign were measured by hand. The ratio of sign to pole was measured to be $\frac{25}{55}$. Therefore, the bottom point of the pole can be found by taking the height of the sign's bounding box, multiplying by the reciprocal of the above proportion, $\frac{55}{25}$, and then adding this number to the bottom middle pixel of the bounding box. Now that the ground-level pixel of the sign is know, H can be used to project the pixel to the car's coordinate frame on the ground.

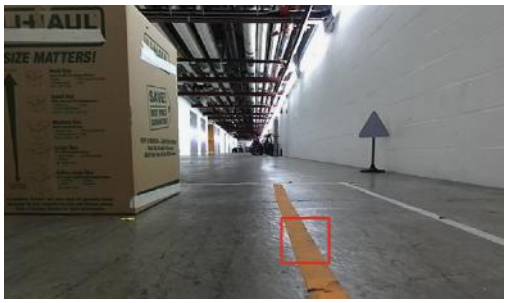
Once the stop sign is detected, a state machine is required to instruct the car of its next action. If a stop sign is detected for the first time, the distance to the stop sign is checked. If the distance is lower than 1.0

meters, a full stop is published to the racecar for a full second. The car's wheels will maintain their previous steering angle as to not throw off the car's path. Upon the full stop being completed, the state variable `self.stop_bool` will be set to true. This variable will only be set to false again once the racecar loses sight of the stop sign. As long as the variable is true, the car will not stop; this logic is implemented so that the racecar will not stop at the same sign more than once.

3.2 Evaluation

3.2.1 Qualitative Results: Reliability of Color Segmentation - OM & JS

Line Detection Evaluation Validation of the color segmentation approach to line detection can be done qualitatively by examining the detected bounding box at various points throughout the city. If the bounding box exclusively encloses the orange road line, ignoring distractors such as the surrounding brown boxes, then the algorithm is successful. Figure 13 shows three such qualitative tests, one in which the racecar is placed on a straight-away, and two in which the racecar was placed before an immediate turn. The second turn test is done in an area with poor lighting in order to evaluate the lower color bound.



(a) Straight-away qualitative test

(b) Turn with poor lighting qualitative test

(c) Turn with good lighting qualitative test

Figure 13: Images visualizing the results of three qualitative tests designed to evaluate the performance of the color segmentation algorithm used to identify the orange road lines. The resulting bounding box is visualized in red. It can be seen that the line is properly identified in all cases, and the brown boxes are not confused for orange, regardless of various lighting conditions.

Stop Sign Detection Evaluation Similarly, validation of the color segmentation for stop sign detection was done by placing the stop sign sign at random locations throughout the city. If the bounding box only encloses the octagon of the stop sign, and estimates the base of the stop sign in the correct location, then our stop sign detection algorithm is successful. Figure 14 shows three examples of qualitative tests that we conducted. First, the stop sign is placed directly in front of the car before a right turn. Second, the stop sign is slightly offset towards the inside of the turn during a right turn. Finally, the stop sign is slightly offset towards the outside of the turn and placed at an angle before a left turn. In each of these cases, the car successfully detected the stop sign and performed a full stop as designed.

(a) Qualitative test for stop sign detection before a right turn

(b) Qualitative test for stop sign detection during a right turn

(c) Qualitative test for stop sign detection before a left turn

Figure 14: Images visualizing the results of three qualitative tests designed to evaluate the performance of the color segmentation algorithm used to detect the stop sign. The bounding box for the stop sign is visualized as a green rectangle, and the base of the stop sign is visualized as a blue circle. The bounding box for the orange road lines are still visualized as a red rectangle. It can be seen that the stop sign is properly identified in all cases, and the brown boxes are not confused with either the red stop sign or the orange road lines.

3.2.2 Quantitative Results: Tuning our PID Controller - AY

To tune and quantitatively evaluate our line-following performance, we track the angle error as we traverse the orange path at varying PID gain levels and compute error graphs along with average errors (see Figure 15). We found empirically that a large K_i actually increased the instability of the controller due to the fact that the problem was more centered around quickly reacting to the sharp turns than reducing steady-state error. So, we eventually determined it was optimal to fix K_i at an extremely low value, decided to be .005. K_p was then tuned at this fixed K_i. It was found that the value of $K_p = 1.6$ works best for optimal turning, which was verified through running trials on the Mini-City. For $K_p < .8$, the car was unable to make the turns required to navigate the city. At the fixed speed of 0.5 m/s, we are able to reach the end of the Mini-City in 13.6 seconds across three trials on average using this proportional gain. The car maintains an average distance of 3 cm from the pursuit point, which keeps our car within the desired distance to the orange line.

4 Conclusion - AY

These two challenges of track racing and city driving highlight the strengths and weaknesses of the two types of approaches. In track racing, we did not have difficulty segmenting out the white lines. The challenge there belonged to filtering out distracting lines and choosing an appropriate pursuit point. Because slope was a key identifier of distractors, Hough lines offered a significantly better approach since they allowed us to leverage geometric information. Additionally, because there was so much noise in the line detection bounding

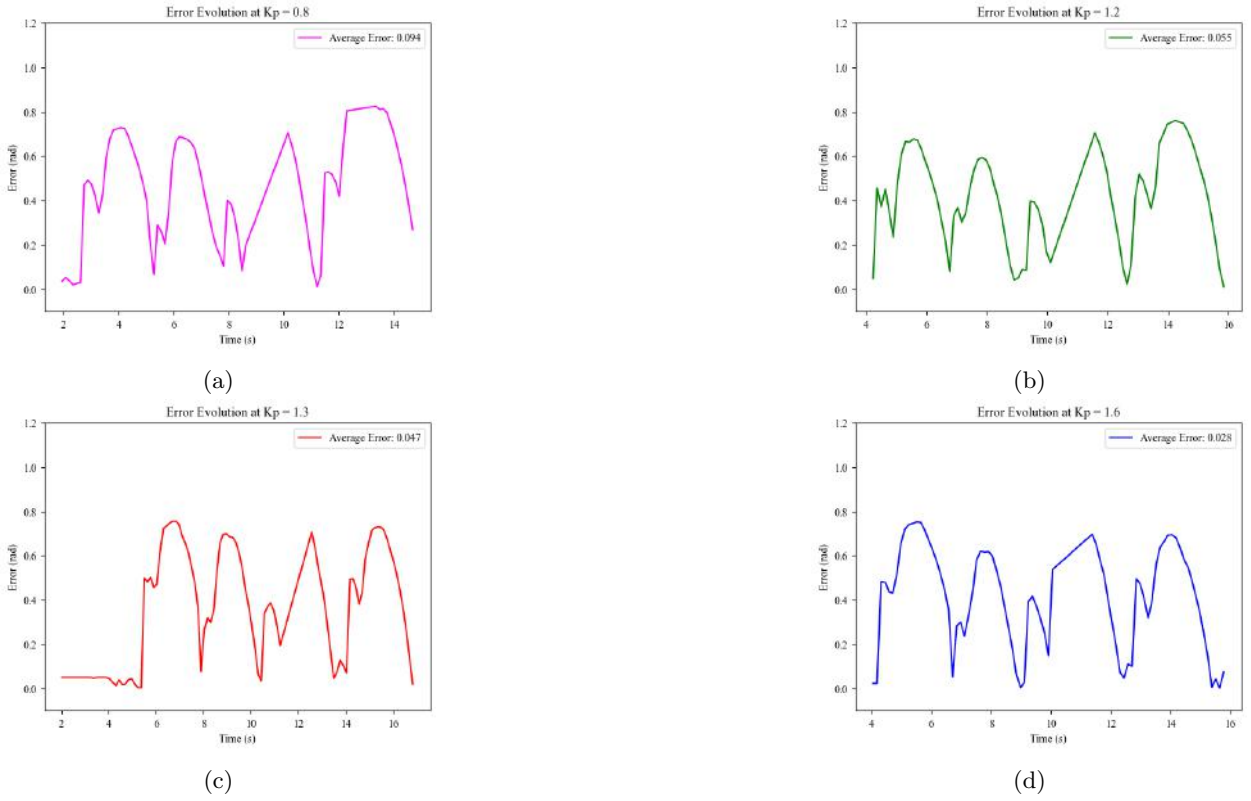


Figure 15: These graphs show the error evolution over time as we traverse the path through the Mini-City at varying K_p values. Values of K_p outside of this selected region had erratic behavior or failed. We see that the value of the gain giving the best average error seems to be $K_p = 1.6$. As always, we expect peaks in the graph to occur at each of the four turns in the path.

boxes, the average point to pursue was unpredictable. This issue was amplified by the fact that there was left-bias in the ZED camera image. The Hough lines circumvented this issue by finding geometric representations of the lines and simply computing the average point between the lines at some fixed lookahead distance.

On the other hand, our Hough line approach did not work well for city-driving. As we have learned throughout the course, it is best to pursue simple solutions when possible. Although our Hough line approach was working well in ideal cases (e.g. car positioned exactly on the line facing the turning corner), many issues would arise with slight perturbations. This seems to be the issue with most complex methods applied to simple methods: they have trouble generalizing to all cases without applying additional logic. Color segmentation turned out to indeed be sufficient for the line-following, where the main difficulty lied in selecting the right bounds for color segmentation. To do so, we leveraged HSV values, which offered more interpretable bounds for colors.

Overall, our work demonstrates the strengths and limitations of these two methods that should be accounted for when tacking any line-following problem. We hope our work provides additional insight into these two approaches and that we can apply this knowledge to future endeavors in this field.

5 Lessons Learned

MW From a technical standpoint, I encountered significant challenges when working with color segmentation, despite the initial problem appearing relatively straightforward. During our tests in Part B, we faced numerous issues that required constant adjustments, right up until the day of the competition. One

particular challenge arose from the similarity in shade between the cardboard boxes and the target line. However, through diligent efforts, we managed to fine-tune the hues and achieve satisfactory color segmentation results. Interestingly, our initial approach utilizing state machines did not yield the desired outcomes. Nonetheless, I was delighted that our team devised a novel combination of PID and pure pursuit algorithms, which proved effective in various scenarios. This experience taught me the value of considering a fusion of multiple approaches when seeking the optimal solution.
From a communication standpoint, this course has provided me with valuable insights into the time demands of robotics. Sometimes hardware or software issues may not go our way so its important to leave enough room to debug and figure out what's wrong. Additionally, talking through algorithms and approaches taught me the importance of communication. Overall, this has been an extremely fun class and look forward to what's new.

AY From a technical perspective, most of this lab relied on experience from previous labs. In Part B, we relied entirely on a segmentation approach from the Visual Servoing Lab. However, it was interesting to solve line following when unanticipated distractors were involved (i.e. the cardboard boxes). We had to pay significant detail to our color bounds as they drew very fine lines to distinguish between boxes and the orange line. One aspect I really enjoyed that was different from previous labs was applying Hough transforms to filter out horizontal lines in Part A. We had learned about Hough transforms in lecture, but before this lab I had not implemented them, and it was interesting to see them work so well for Part A. Although they didn't work for Part B, I was very satisfied with my ability to understand and implement them quickly in a new setting.
From a CI perspective, this lab was by far the most difficult to manage in terms of time. Back when we started the challenge, in order to stay on track, I set milestones for every few days specifying what to complete by that date. This was crucial as I had three other big final projects to complete, and I found this planning extremely helpful for staying organized with so many things on my mind. It was also really helpful to have someone keep me accountable; Owen and I established a routine to go to the track every night at 8 and work until closing in order to finish Part A. We had a similar routine for Part B, although we ended up scrambling to finish it due to some unexpected delays in Part A. Despite the technical challenges (and sometimes frustration), it was enjoyable to work through solutions together and encourage each other along the way. Overall, I am really happy with how things worked out (getting things done on time, the quality of work, etc.) and I am really proud of Owen in particular for working with me many tireless nights in the weeks leading up to the final challenge.

JS Technical: I learned that it is very frustrating to work with color segmentation. As soon as the cardboard boxes were added, we ran into trouble distinguishing between the orange line and brown boxes. This required a lot of tuning on our part, and seemed to change day to day depending on the mood of our robot (just kidding, it was most likely due to small changes in lighting and the positions/orientations of the boxes). Dylan mentioned setting bounds for color segmentation using machine learning trained on pictures labelled manually, which is a very interesting idea and could be promising if we had a little more time. As for the controller, this lab was largely based on previous labs. However, it was very cool that my teammates thought to combine PID and pure pursuit to create the optimal controller. I never would've considered combining different control algorithms, and it ended up working very well, even at high speeds. Overall, this class taught me to consider all types of solutions, whether it be the most basic approach or a combination of different approaches. Communication: Through this course, I learned that robotics always requires more time than expected. Because it is such a multidisciplinary field, there are bound to be errors in hardware, software, the connection between the two, or both at the same time. When debugging, I observed that talking out loud through the logic of the algorithm and/or drawing things out helped tremendously in thinking clearly and finding solutions. The presence of a teammate to work through these errors together is also very beneficial to maintain sanity.

OM One of the biggest things I took away from this class is that each technical problem should be approached with an open mind. Going into a problem with the idea that it seems simple or similar to a previous problem you've seen can lead to bias in problem solving and lead one down dead ends. For example,

for part B, Alan and I believed that by skipping over color segmentation and moving right to Hough lines, we would be saving time. We had this bias because that is what happened during our part A process; however, the Hough implementation for part B ended up being far too complex and contrived for the time we were given. In the future, I will do my best to start from scratch for each problem and keep an open mind during the problem solving process, while using previous approaches as mere inspiration.
From a communications perspective, I learned to persevere in the face of sub-optimal working conditions. When working through part B, Alan and I were very low on time, as very small changes in lighting or arrangement of the city were throwing off the behavior of the robot from day to day. The Friday before the challenge, we spent almost the whole day increasing the robustness of the controller to work in varying environments. As we were very short on time to complete the stop sign extra credit, we decided to return the morning before the challenge after what was essentially a nap. I realized that I wanted to do work I was proud of, and I wanted to see the development of the full controller to the end. It was really satisfying to work through the challenge each step of the way and devote so much time to this project. When the final challenge was over, although our robot performed very well, I was more proud of the way we persevered up to the final moments.

RS From a technical perspective, Part A of this lab was all about figuring out how to modify our line follower code from Lab 4 to meet the demands of lane following. It was an interesting problem, and I enjoyed coming up with the idea of using color segmentation on the left and right for the white lines and then averaging them to create the line we're supposed to follow down the center of our lane. Also, on Part B, I had originally thought that we would use path planning, for that way, we would be able to try the extra credit task of parking at another portal. I spent a lot of time working out the state machine we would have used. However, it unfortunately went unused because the Stata basement layout was distinct from the map we had, and this would have made it impossible for us to properly localize our position within the map. I also worked out the state machine for stop sign detection as well. Overall, the problems were quite interesting and required lots of thought, so I enjoyed the technical aspects of this lab.

From a communications perspective, this lab taught the importance of balancing my workload and approaching problems from the correct angles. With multiple final projects, I have been stretched very thin, and I would try to do as much theory as I could, for theory is something that you can come up with in the back of your mind as you do something else. Moreover, it ensures you don't waste time pursuing dead ends. Without doing that, this project would have been all the more unmanageable.

SY On the technical front, the complexity of this project lay in the effective integration of various algorithms into a functional whole and the application of this system to real-world scenarios. The transition from theoretical constructs to practical implementation was not a straightforward process, demanding immense patience and a significant investment of time for debugging and adjustments. The constant variations in the test field lighting meant that our color detector had to be carefully and continuously fine-tuned, avoiding to mess up with brown tape and lines. The real-world environment presented unique challenges, which couldn't always be addressed with high-level algorithms like path planning. We ultimately decided to employ simpler methods, specifically PID and pure pursuit algorithms. While these methods are easier to implement without inducing unpredictable applying issues, their tuning required considerable effort due to their lack of robustness. This experience emphasized the need to carefully consider various factors such as environment, tasks, timeline, and available resources to determine the most suitable approach for a project.

From a CI perspective, this lab highlighted the importance of effective time management, particularly when every team member juggling multiple final projects. The question to be answered was: what is the most effective contribution one can make given limited time? Each member of our team demonstrated their answer to this through contributions, whether it was theorizing, coding, tuning, or evaluating. Through collective effort, we navigated through multiple iterations of theory and practice, ultimately achieving our goal, in this challenging journey. So, good jobs, and a big load of love to we MAJORS.