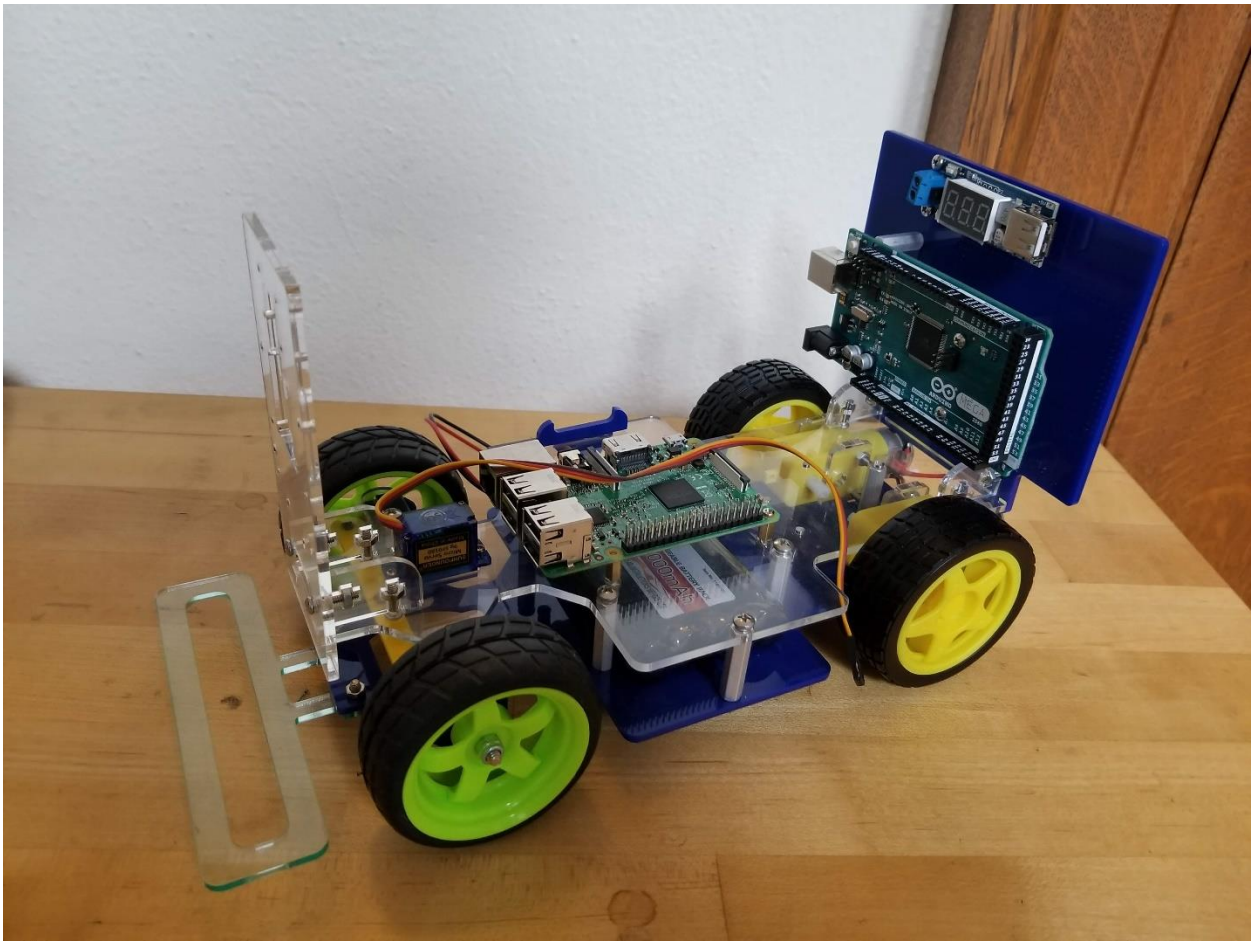
Autonomous Car Robot



Shane Shearman

Jacob M. Abel Summer 2018 Internship Report

Mechanical Engineering and Applied Mechanics

University of Pennsylvania

1 Overview

The Fall 2018 introduction of the class Control for Autonomous Robots (ESE 421) to Penn Engineering meant students would have the opportunity to learn about the control mechanisms, design and software of an autonomous vehicle. There was, however, limited implementation of the projects designed for this course at the beginning of Summer 2018.

In order to allow for the class to be run efficiently, it was necessary to uncover any potential challenges and how to address them; whether related to the car itself or the logistics of the class.

The aim of this research was to explore:

- Possible improvements to the mechanical design of the robot
- Computer vision and neural network execution techniques
- Any logistics necessary to ensure the class ran smoothly

This allowed for various avenues of engineering to be investigated including computer vision/image processing, deep learning, mechanical testing, Simulink and 3D modelling; all of which will be detailed in this report.

2 Mechanical Design Updates

Several design changes were made to the previous version of the car to allow for better performance, packaging and dependability. Each of these additions utilizes materials and fasteners readily available in Penn's engineering quad to allow for convenient and expeditious replacement of components throughout the semester. These changes include:

- Addition of mounting for Pi computer, Pi cam, GPS, gyro

- Repositioning of Arduino to allow for better wire routing and space for GPS and gyro chips
- Strengthening/bracing of parts with previously high stress concentrations

The updated robot is fully and accurately represented in CAD to allow students to make design changes as necessary. The inclusion of all fasteners in CAD also means students can use SOLIDWORKS as a guide during car assembly. Finally, there is the ability to generate a BOM (Bill Of Materials) so that future classes can conveniently order components.

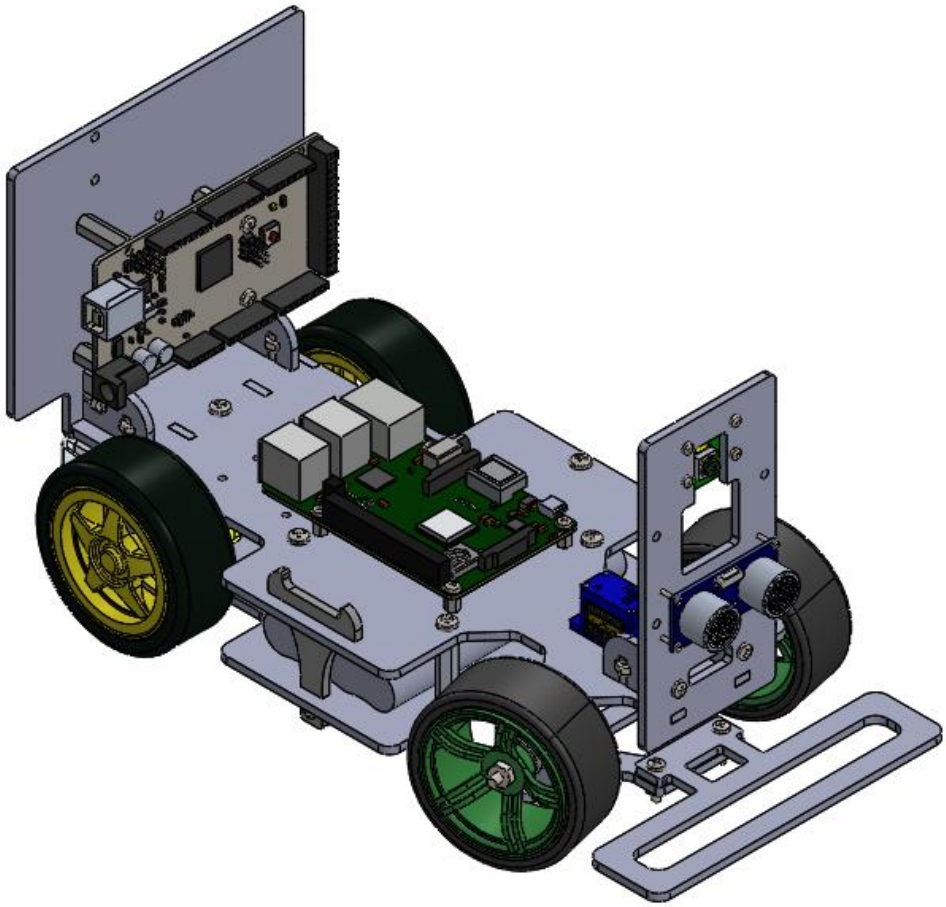


Figure 1.1: Full Robot CAD

3 Simulink Model

One of the projects for the class is to have the robots follow each other autonomously in the form of a convoy/platoon. Within a platoon, the manner in which the preceding car interacts with those behind it offers up many interesting physical phenomena. The velocity, acceleration, jerk and steering angle all respond more extremely as one moves toward the back of a platoon. This interesting question of how P.I.D. controls can be utilized to curtail the aforementioned extremities may be investigated in Matlab via a Simulink model.

Thus, models were synthesized to predict the behavior of the motor and steering angle interactions between successive cars as well as that of the overall platoon. An overall model will be eventually available to the class to improve upon or make changes as necessary as the actual physical performance of the car is observed.

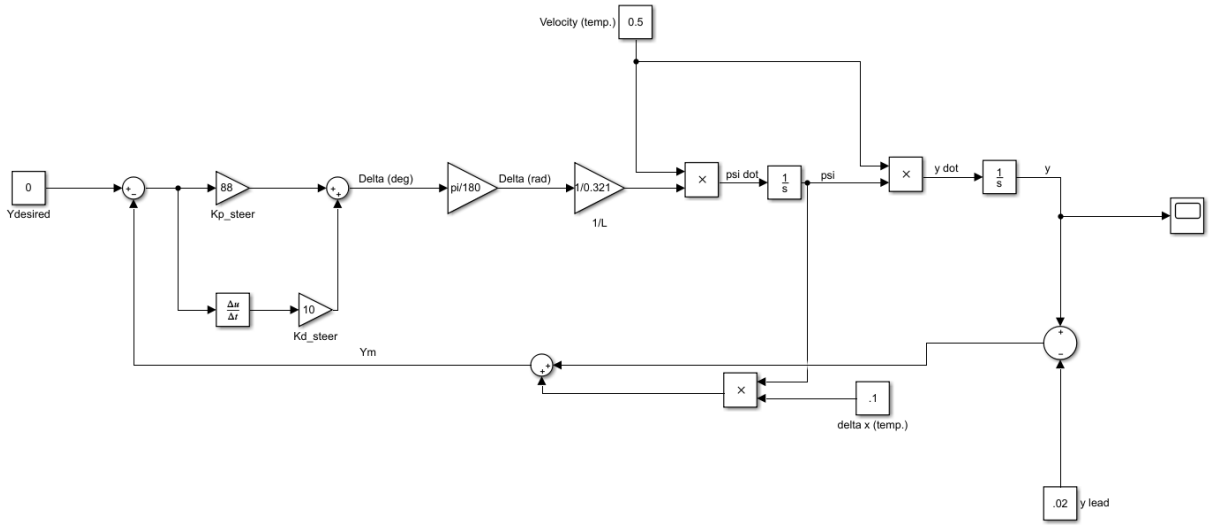


Figure 3.1: Steering Model

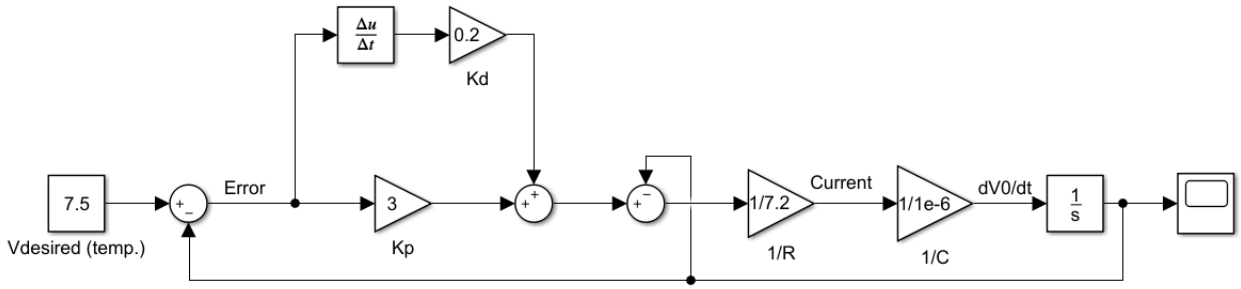


Figure 3.2: Motor model

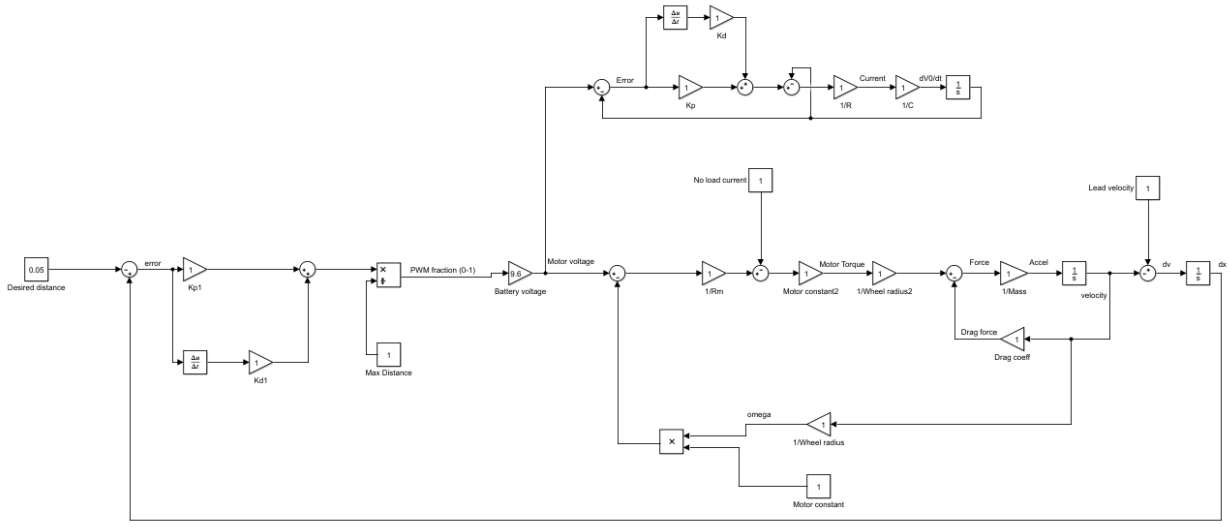


Figure 3.3: Full Platoon model

4 Image Processing

A large part of the class involves using computer vision for image processing and data acquisition. To achieve this, OpenCV was installed on a Raspberry Pi with a Pi Cam used to capture images and video for analysis. This Pi coupled with an ultrasonic sensor for distancing and I2C communication with an Arduino Mega allows for autonomy in both class projects: platooning and Penn Park driving.

4.1 Platooning

In order to accurately follow the robot in front of it, it was necessary to have a "target" for the trailing car to view. For this purpose, a red acrylic circle was fastened to the rear of each car and sample images were taken from a Pi Cam for processing in Python.

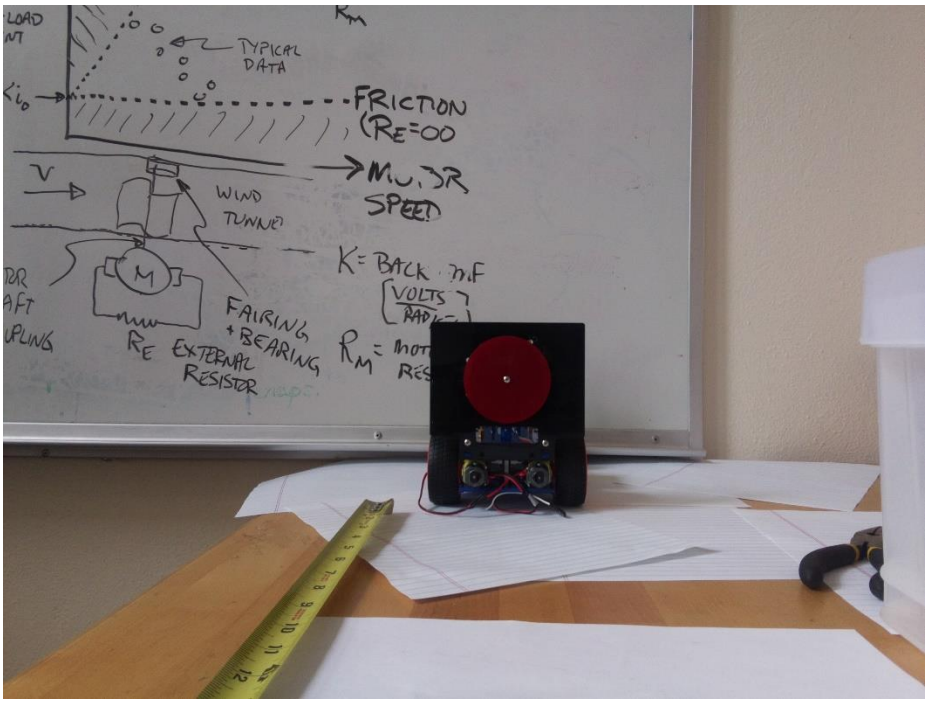


Figure 4.1.1: Rear Car image

This RGB (Red, Green, Blue) image was converted to the HSV (Hue, Saturation, Value) color space to create a better contrast between red and the background colors.

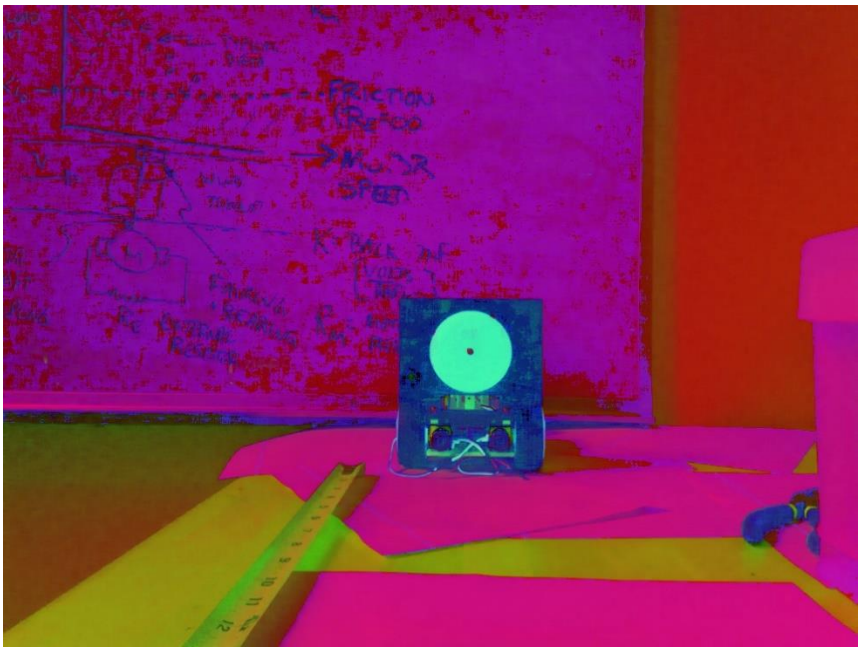


Figure 4.1.2: HSV image

From this point, a mask was applied to isolate the target pixels by turning them to white and making all other pixels black. A Gaussian blur was also applied to assist the software with edge detection.

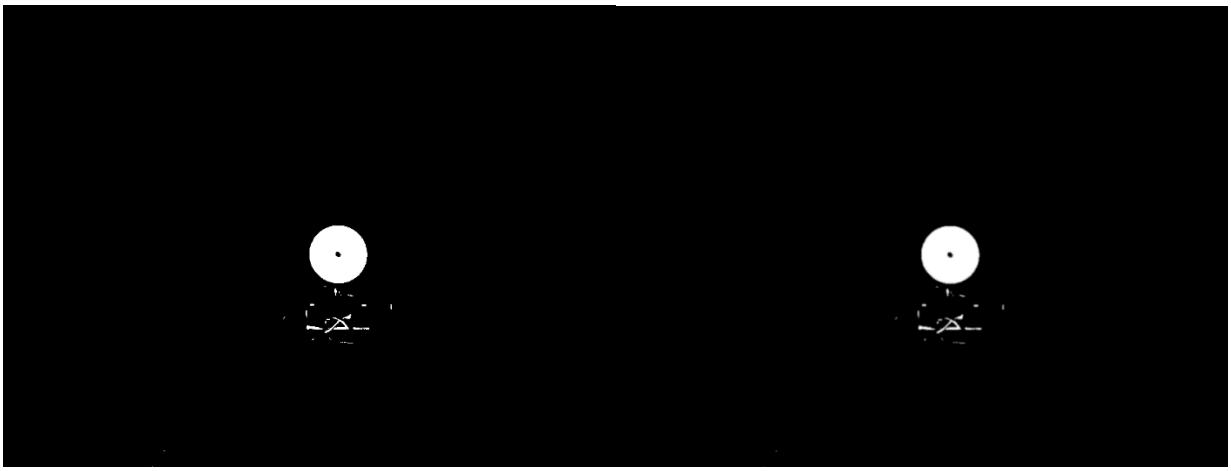


Figure 4.1.3: Masked image

Figure 4.1.4: Blurred image

Canny edge detection was then used to outline the target.

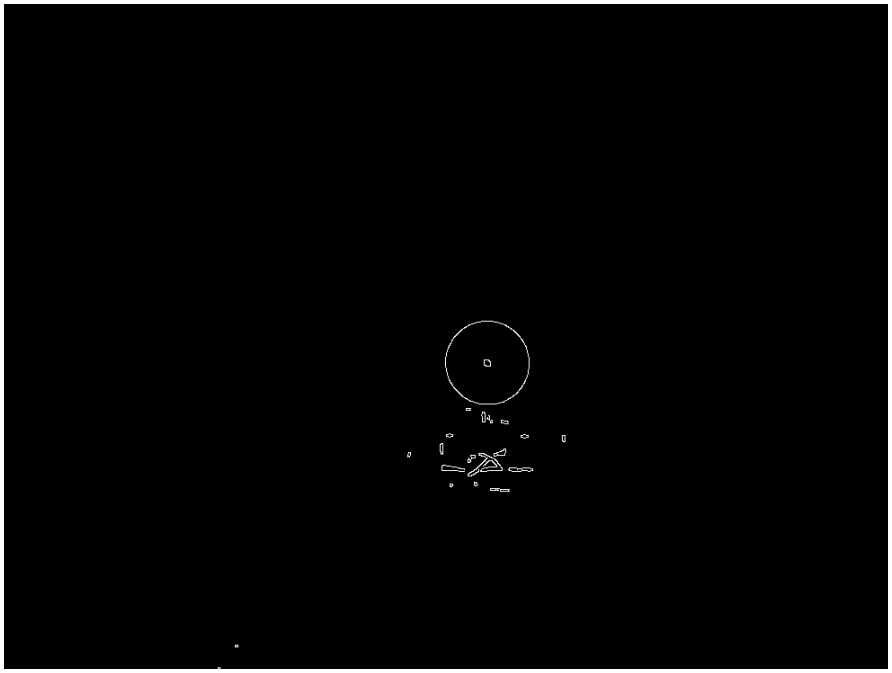


Figure 4.1.5: Canny edge image

As it can be seen in the images above, the red from the motor wires as well as a couple other components were also detected. To ensure these were ignored, an algorithm was used to search for the largest contour and put a box around it to confirm this is what is desired. Using the focal length of the Pi Cam as well as the diameter of the target circle, it is possible to calculate the distance between cars.

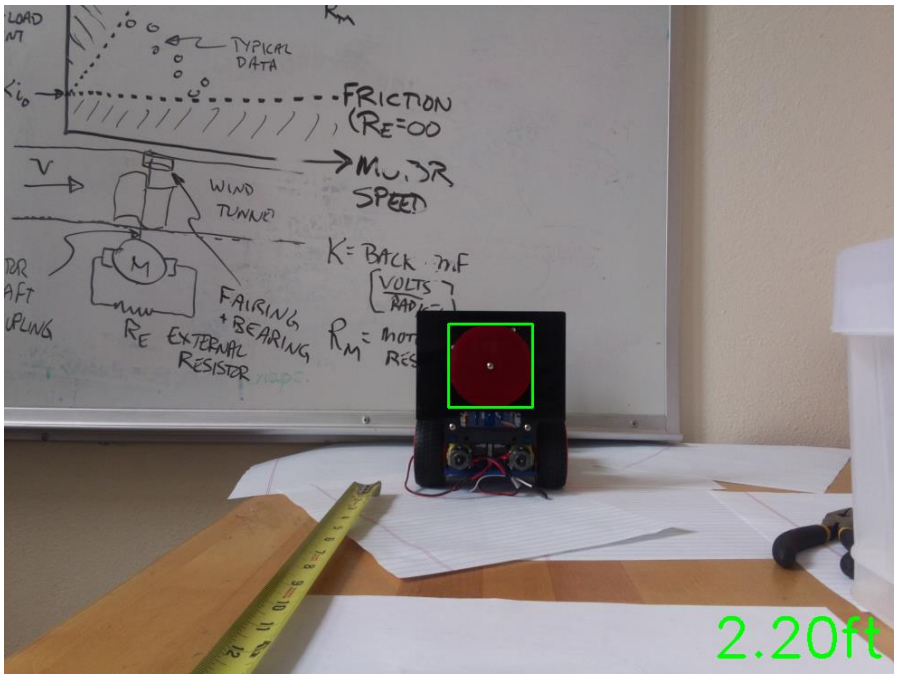


Figure 4.1.6: Detected target with distance image

From here, the offset of the box center from the image center could be used to determine steering angles and an ultrasonic sensor supplemented by the software's distance calculation is used to determine motor velocity (or PWM); both making use of P.D. controls.

4.2 Penn Park Driving

To determine how to approach the problem of autonomous driving in Penn Park, it was important to know what the car camera would see. To achieve this, a car was manually driven around the park while the Pi cam captured more than two hundred images. The image processing for this is similar to that of platooning with a few exceptions.

Figure 4.2.1: Sample park image

Because the trees would provide an unnecessary complication in edge detection and subsequently controls, a region of interest was defined for each image. These cropped images were also made into plots so that coordinates could be easily found on each image.

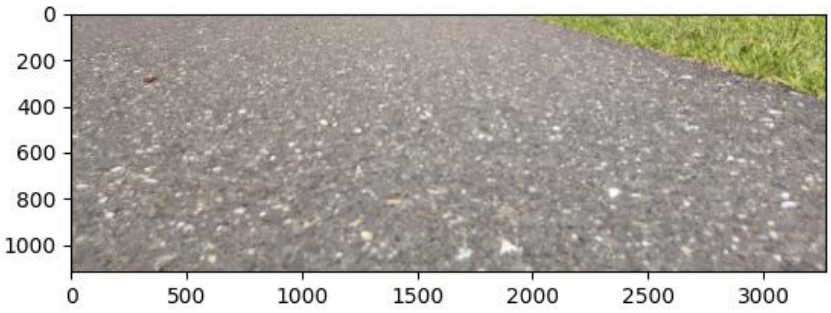


Figure 4.2.2: Cropped image

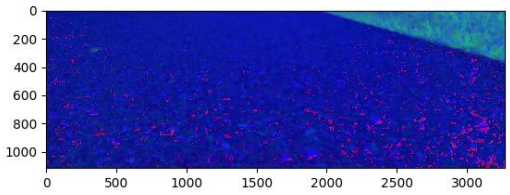


Figure 4.2.3: HSV image

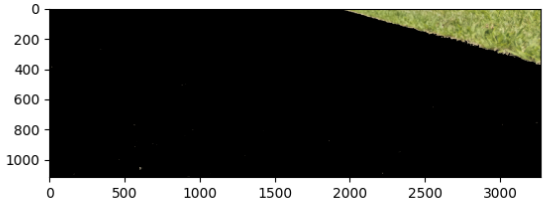


Figure 4.2.4: Mask image

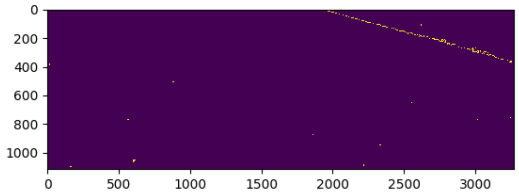


Figure 4.2.5: Canny edge image

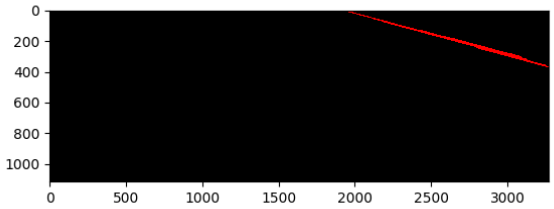


Figure 4.2.6: Hough lines image

As seen in image 4.2.6, a Hough transform (probable) algorithm was applied to the canny edge image to produce a line for the car to follow. The final image appears as:

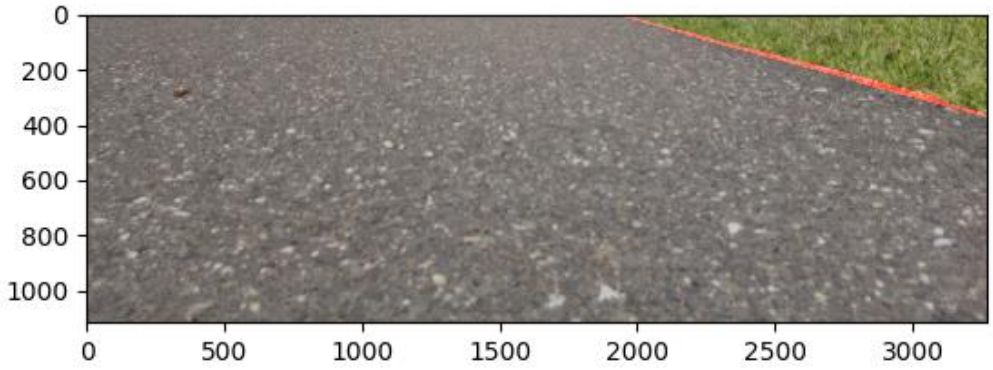


Figure 4.2.7: Final processed image

Using the slope of this line and its point of intersection/angle with the image center, it is possible to determine the steering angle of the car as well as the distance from the edge. With this data supplemented by the GPS location of the car, it is possible to navigate Penn Park autonomously.

5 Conclusion

Overall, this project can be labelled a success. Mechanically, the car is well equipped to traverse Penn Park with its sturdy design. It can also be easily assembled, fixed or altered due to the choice of materials, components and availability of CAD.

Furthermore, there now exists software foundations in Python and OpenCV for students to reference as they approach image processing in their own way. Once the robots are functional, students can also explore their behavior in Simulink.

While not explored extensively during the summer, there is room for the class to explore deep learning neural networks such as Google's Tensorflow for live object detection and path planning in the future.