



MAY 3, 2019

PROJECT REPORT: AUTONOMOUS NAVIGATION ON ROAD

AUE 8930: AUTONOMOUS DRIVING TECHNOLOGIES

NAYAN DESHMUKH



Executive Summary

Academic project on 'Autonomous Navigation on Road' was done and the technical approach and experimental results are recorded in this report. The report consists of four sections – Problem Statement, Autonomous Lane Tracking Control, Road Sign Recognition and Vehicle Reaction, Project Achievements and Project Challenges.

A detailed technical approach and experimental results are discussed for the autonomous lane tracking control and road sign recognition and vehicle reaction.

Table of Contents

Executive Summary	1
List of Figures	3
List of Tables	3
Introduction to Autonomous Vehicles.....	4
1. Project Statement	4
2. Autonomous Lane Tracking Control	5
2.1. Technical Approach.....	5
2.1.1. Mount for front camera	6
2.1.2. Camera Calibration	6
2.1.3. Edge Detection.....	6
2.1.4. Define Region of Interest Polygon	8
2.1.5. Hough Transform	8
2.1.6. Post Processing	10
2.1.7. Inverse Projection	12
2.1.8. Lane Tracking Controller	12
2.2. Experimental Results	13
2.2.1. Summary of Lane Detection Results	13
2.2.2. Experimental Results of Lane Tracking Control	15
3. Road Sign Recognition and Vehicle Reaction.....	19
3.1. Technical Approach.....	19
3.1.1. Create training data	19
3.1.2. Label Images.....	19
3.1.3. Train R-CNN Model	20
3.1.4. Trained R-CNN Model to Identify Road Signs	20
3.1.5. Computer to Computer Communications	20
3.1.6. Vehicle Reaction Control.....	21
3.2. Experimental Results	23
3.2.1. Road sign recognition, Communication and Vehicle Reaction Results.....	23
4. Project Achievements	25
5. Project Challenges.....	25
References	26
Course Evaluation	27

List of Figures

Figure 1: Two white lines depicting road lanes	4
Figure 2: Road signs	5
Figure 3: Lane tracking control process flow	5
Figure 4: Front camera mounting	6
Figure 5: Camera calibration using MATLAB camera calibrator app	6
Figure 6: Region of interest polygon	8
Figure 7: Problem with Lane detection	10
Figure 8: Measurement noise covariance	11
Figure 9: Stabilization of lines due to Kalman filter	12
Figure 10: Edge detection using canny algorithm	13
Figure 11: Region of interest polygon	13
Figure 12: Hough transform	14
Figure 13: Stabilization of lines due to Kalman filter	14
Figure 14: Limit steering angle	15
Figure 15: Missing right line	15
Figure 16: Solution to missing line problem	16
Figure 17: Value of e_{fa} for one lap on the test track	17
Figure 18: Variation in departure angle for one lap on the test track	17
Figure 19: Steering input to the steering servo from the controller	18
Figure 20: Process flow for road sign recognition and vehicle reaction	19
Figure 21: Training images of the two road signs	19
Figure 22: Road sign detection	20
Figure 23: MATLAB code to quantify road sign recognition	20
Figure 24: UDP communication layout	21
Figure 25: Division of map of test track	21
Figure 26: Vehicle Reaction Control	22
Figure 27: Vehicle reaction control explained	22
Figure 28: Varying value of data with time samples	23
Figure 29: Varying value of 'state'	24
Figure 30: Vehicle reaction control	24

List of Tables

Table 1: Iterations with canny parameters	8
Table 2: Iterations with Hough peaks	9

Introduction to Autonomous Vehicles

Autonomy has become one of the prime research areas in recent times. Autonomy finds its applications across all the engineering disciplines and promises a revolutionary transformation of almost all the industries of the present world. Automotive industry is one of the leading industries which is seeing a paradigm shift in its way of solving complex real-world problems with the help of autonomy. Autonomous driving technologies promise road safety, effective use of time, reduced traffic congestion, reduction of space required for vehicle parking and low fuel consumption. Hence, it is one of the hot topics in the automotive industry currently.

An autonomous vehicle combines a variety of sensors to perceive their surroundings, such as RADAR, LIDAR, Ultrasonic sensors, GPS, camera, speed sensor, acceleration sensor, attitude sensor etc. Advanced control systems interpret sensory information to identify appropriate navigation paths, as well as obstacles and relevant signage. Some of the autonomous driving functions incorporated in modern vehicles are autonomous navigation, adaptive cruise control, lane tracking and switching, autonomous parking, V2X communications to name but a few. All these functionalities developed due to the exploration of autonomous technologies aim at bettering the quality of human life with the help of technology. But autonomous vehicle technologies also face some challenges which constraint full utilisation of the autonomous driving capabilities of the vehicle. Winning public and individual trust, unfavourable government regulations, hardware/software reliability, loss of privacy and threat of security are few challenges that autonomy faces currently. Nevertheless, society is changing continuously, and predictions show that the socio-economic conditions would be favourable for autonomous vehicles and the number of autonomous vehicles would go on increasing with each passing year. The rapid advancements in autonomous driving technologies is attracting more and more engineers to work in this field.

An academic project was done on vehicle's 'Autonomous navigation on road' in which the vehicle was programmed to track lanes using camera and recognize road signs using deep learning. This report documents the procedure followed to program the vehicle for lane-tracking controls and road sign recognition and discusses the findings and results of the project.

1. Project Statement

The project on 'Autonomous Navigation on Road' has four main tasks – first, autonomous lane keeping, in this task, the vehicle has to maintain its trajectory within two white lines as shown in figure 1. A webcam mounted at the front of the vehicle continuously senses these two white lines and controller controls the steering servo as per the sensory inputs.

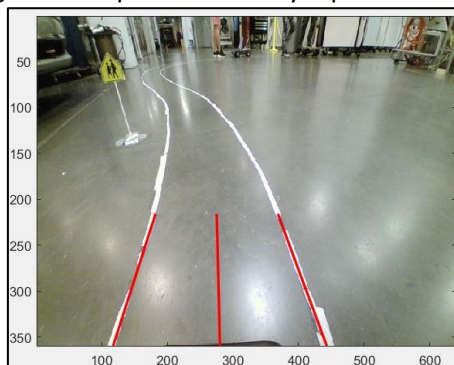


Figure 1: Two white lines depicting road lanes

Second, road sign recognition, with the use of one more camera on the vehicle, it must autonomously recognize school sign and stop sign shown in figure (2).



Figure 2: Road signs

One computer was used to receive the sensory data from the front camera that was detecting road lines and another computer was used to receive sensory data from the side camera that was sensing the road signs. Next task was to send road sign information from second computer to the first computer through Wi-fi communications. This was done using UDP communication protocol.

The last task was to code for controlling the vehicle as per sensory data available from both the cameras. According to the lane detection data from the front camera, a controller was to be designed so as to control the steering servo of the vehicle to maintain the trajectory of the vehicle within the two white lines representing the road lanes. The vehicle has to stop at the stop sign for two seconds and then run at a high speed. On seeing the school sign, the vehicle has to reduce its speed by half.

2. Autonomous Lane Tracking Control

2.1. Technical Approach

Process Flowchart

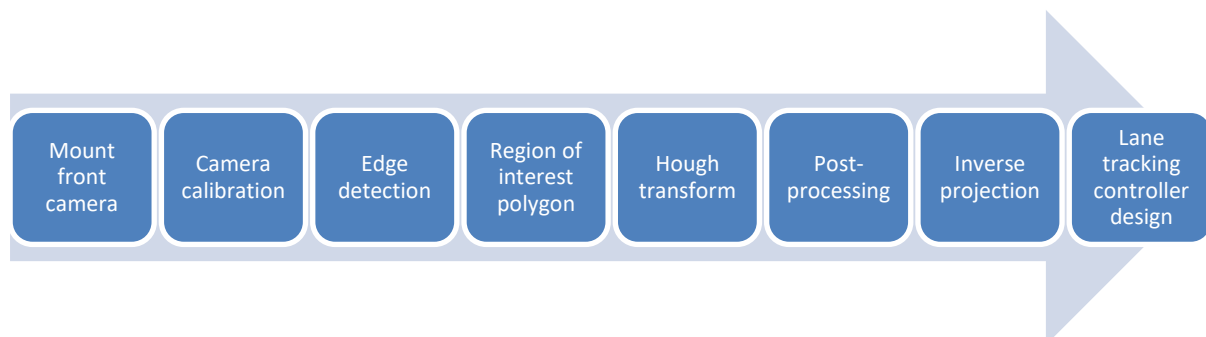


Figure 3: Lane tracking control process flow

2.1.1. Mount for front camera

The very first step in lane detection is mounting of the camera. A webcam was mounted at front center of the vehicle. The camera mounting involved some trade-offs, mounting the camera at a short height from the ground traced the white lines perfectly but it was difficult to run the vehicle even at a moderate speed since the camera only sensed the lanes up to a short distance ahead of it. Mounting the camera high from ground made segregating left lines and right lines difficult and hence, reduced the ability to track the lane properly. Thus, some iterations were done to find an optimum location of the camera so as to get best of both worlds, i.e. good lane tracking as well as increased range of sight of camera. After some iterations, we got an optimum location for mounting the camera which is shown in figure below.

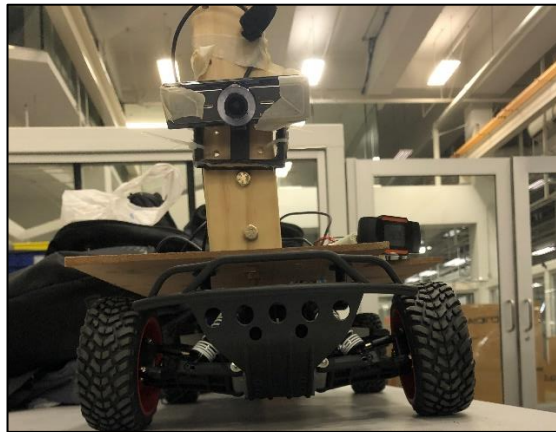


Figure 4: Front camera mounting

2.1.2. Camera Calibration

Camera calibration is important in this project to obtain intrinsic parameters of the camera which would then be used in the controller design. MATLAB camera calibration application was used to calibrate the camera. Some of the photos captured during camera calibration are shown below.

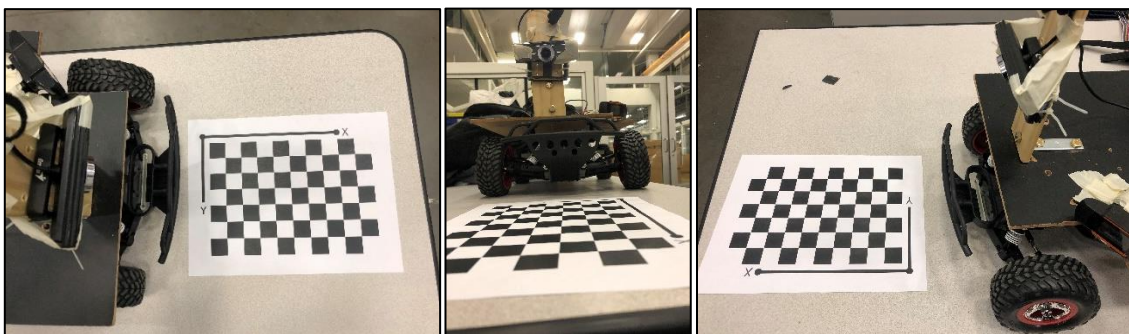


Figure 5: Camera calibration using MATLAB camera calibrator app

2.1.3. Edge Detection

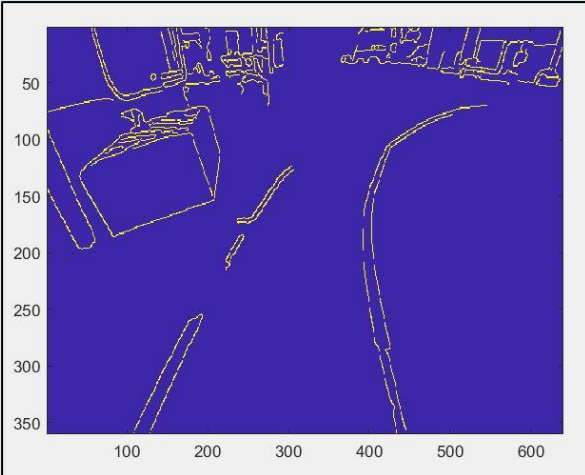
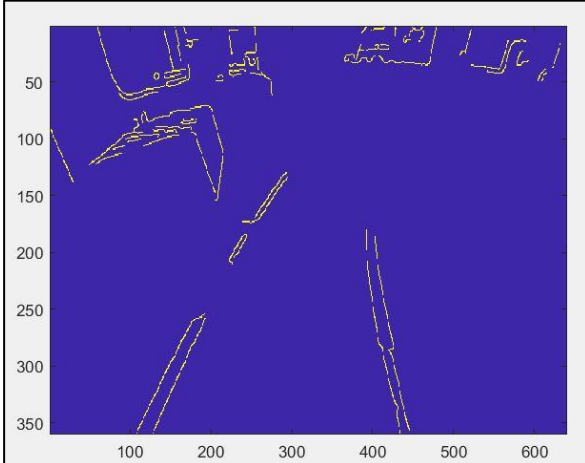
Edge detection is one of the critical processes of the lane tracking control. Correctness of lane tracking control is determined by the effectivity of the edge detection algorithm. In this project, we have used canny edge detection algorithm to detect edges in the image. Canny edge detector is an edge detection algorithm that uses a multi-stage algorithm to detect a wide range of edges in images. The process of canny edge detection algorithm takes place in five steps ^[1] –

- Application of Gaussian filter to smooth the image in order to remove the noise
- Determine the intensity gradients of the image
- Get rid of spurious response to edge detection by applying non-maximum suppression
- Apply double threshold to determine potential edges
- Finalize the detection of edges by suppressing all the other edges that are weak and not connected to strong edges

In this project, we have used inbuilt MATLAB canny algorithm to detect edges in the image because of its good and reliable detection. Iterations were done with the parameters of canny edge detection in order to get the edge detection right so that the algorithm detects optimal number of edges in the image, not more and not less. The MATLAB command for edge detection with Canny algorithm is –

edge_pic = edge (gray_pic, 'canny', [0.2,0.36])

Some of the iterations done with the canny parameters are tabulated below.

Edges detected in the image	Canny parameters
	[0.1, 0.36]
	[0.3, 0.5]

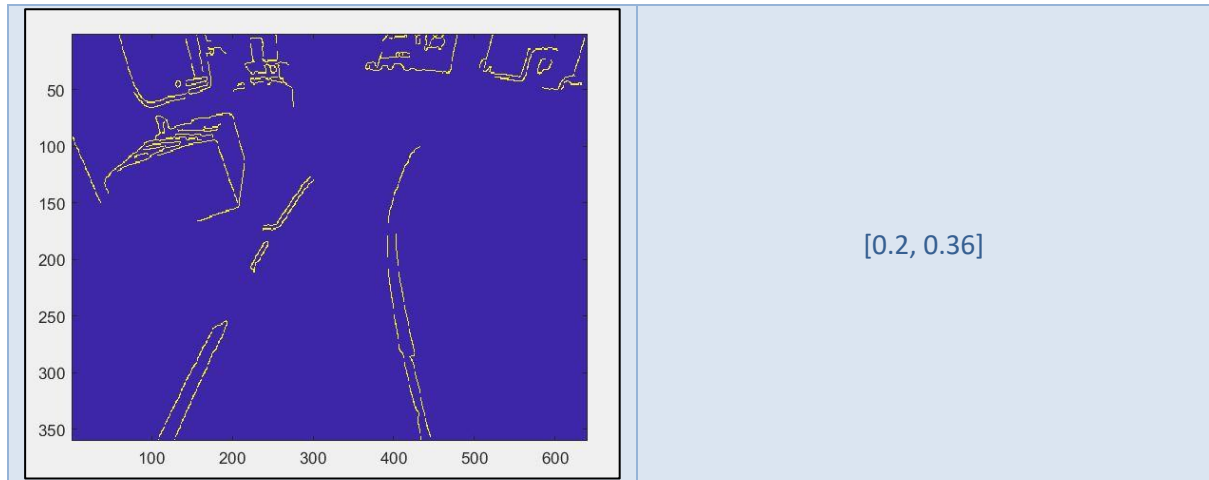


Table 1: Iterations with canny parameters

The last iteration is the one we finalised because it gave optimum number of edges in an image. The purpose of edge detection is to extract useful structural information from different vision objects and dramatically reduce the amount of data to be processed. Hence, we concluded that by keeping canny parameters at [0.2,0.36] we were able to extract only as much that was necessary for lane detection given our region of interest polygon.

2.1.4. Define Region of Interest Polygon

Region of interest polygon was defined to cut out a shape from the image that consisted of lane edges which will be important for lane tracking. Figure (6) shows the shape of the region of interest polygon.

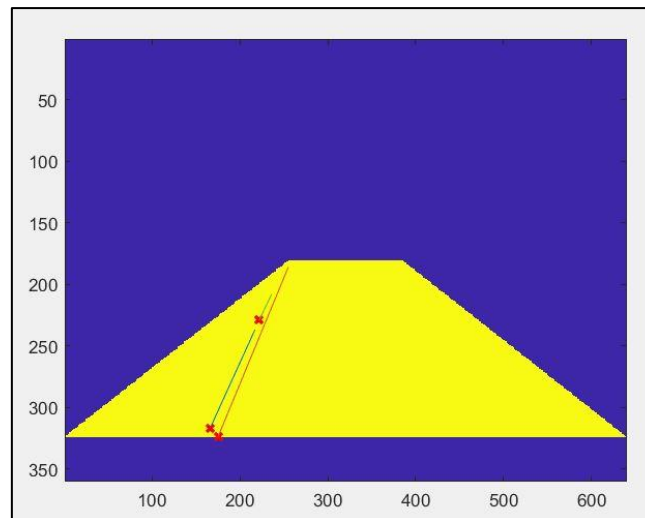


Figure 6: Region of interest polygon

2.1.5. Hough Transform

The Hough transform is a feature extraction technique used for image analysis and computer vision. Hough transform converts geometries of object from image space to Hough space. In image plane, a line which is represented by $y = mx + c$ is plotted as x versus y . On converting the line to Hough space, the same line is plotted as m versus c . Thus, in this project, the Hough transformation was used to convert the edges obtained in image space to points in Hough space and then lines were drawn using these points.

MATLAB command for Hough transform was used to obtain Hough lines. The MATLAB code used is shown below –

```
[H,T,R] = hough(BW);
P = houghpeaks(H,8,'Threshold',ceil(0.3 * max(H(:)))));
Lines = houghlines(BW,T,R,P,'FillGap',6,'MinLength',2);
```

Here, BW is binary image and Hough transform is done on this binary image. We did some iterations with the number of Hough peaks to find a number that was working good for us. The command 'houghpeaks' in MATLAB picks up top k lines (in our case k = 8) that have the highest number of points. The iterations done with number of Hough peaks are tabulated below –

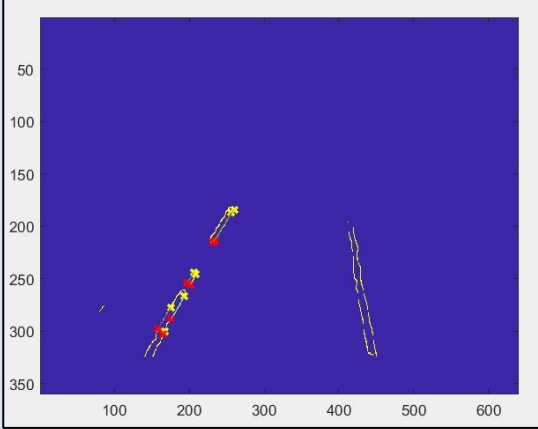
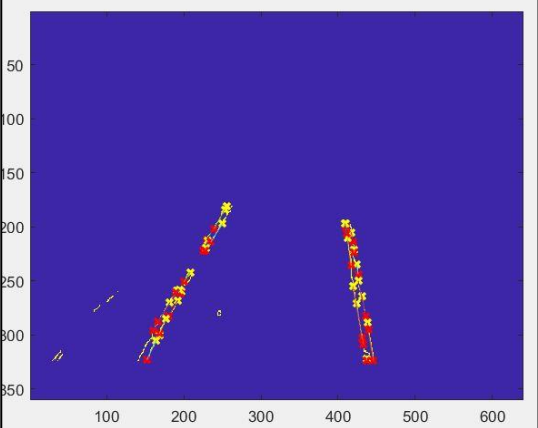
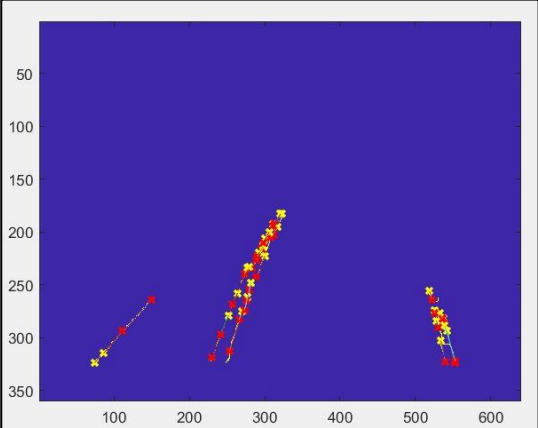
Iteration Result	Number of peaks
	2
	8
	15

Table 2: Iterations with Hough peaks

From table (2), we can see that reducing the number of peaks reduces the number of lines getting selected since 'houghpeaks' picks up top k number of lines that have the highest number of points. Thus, reducing the number of Hough peaks, made it difficult sometimes to track both the lines representing a lane as we can see in table (2). Increasing the number of Hough peaks also had a drawback that even unwanted edges were detected, which reduces the possibility of correct lane tracking. The last iteration in table (2) shows the drawback of having excess peaks. Thus, a balance was required to get a good lane tracking. With number of houghpeaks set to 8, we got a good lane tracking, iteration 2 shows the result of it in the table above.

2.1.6. Post Processing

After obtaining the lines in Hough space, post-processing was done to segregate the lines as right lines and left lines. This segregation was done based on the position and orientation of the lines. To merge a group of lines into a single line, MATLAB 'polyfit' function was used.

One of the problems faced in lane detection was that the detected left and right lines were not stable enough which could have caused undesirable steering behavior and thus, inaccurate lane tracking. Figure (7) shows this problem of fluctuating right lines.



Figure 7: Problem with Lane detection

This problem of unstable lines was resolved by using Kalman filter. Kalman filter was applied to the top point coordinates of both the lines since those points fluctuated a lot as compared to the points at the base of the lines.

The MATLAB code for Kalman filter used for left lines is given below –

```

L_p = draw_lx';
P_pl = Pl + Q;
Kl = P_pl * ((P_pl + RR)^(-1));
draw_lx = polyval(PL, draw_y);
L_m = draw_lx;
draw_lx = L_p + Kl * (L_m' - L_p);
Pl = (1 - Kl) * P_pl;

```

Here,

L_p = Prediction of x and y coordinates of the point for next step

$draw_{lx'}$ = Previous state array of x and y coordinates of the point

P_{pl} = Covariance prediction

Q = Process noise covariance

L_m = Measurement

RR = Measurement noise covariance

Kl = Kalman gain

$draw_{lx}$ = Corrected current state

Pl = Corrected current state error covariance

In the first step, the state (coordinates in our case) of the points at previous timestep was used to predict the coordinates of the point at next timestep. Similarly, the error covariance for the next timestep is predicted using error covariance of previous timestep and adding effect of process noise covariance. In the next step, Kalman gain was found as shown in the code. The measurement noise (RR) was calculated by keeping vehicle's position and orientation fixed and recording the x and y coordinates of points on left lines. The recorded data of x and y coordinates is shown in the figure below and covariance of the data is found in Microsoft Excel.

x	y
245.8897	209.581
245.8488	209.6515
245.8581	209.6346
246.2118	209.5917
246.2401	209.5169
246.3577	209.4199
246.3173	209.4455
246.1581	209.5858
246.2172	209.546
246.3173	209.4455
245.2273	210.1347
245.1966	210.1622
245.2273	210.1347
246.3173	209.4455
0.190188	0.073

Figure 8: Measurement noise covariance

After calculating the Kalman gain and obtaining the actual measurements i.e. actual coordinates of the points, we proceed to the correction step. In correction step, the state of the lines is corrected by comparing the predicted and measured data and error covariance is corrected using Kalman gain and prediction of error covariance.

A similar Kalman filter was designed for the right lines. On application of Kalman filter on both the lines, we observed that stability of the two lines improved dramatically. Figure (9) shows the role of Kalman filter in stabilizing the two line. Both pictures were taken one after the other, one with Kalman filtering and the other without, at exact same point on the test circuit and at same speed.

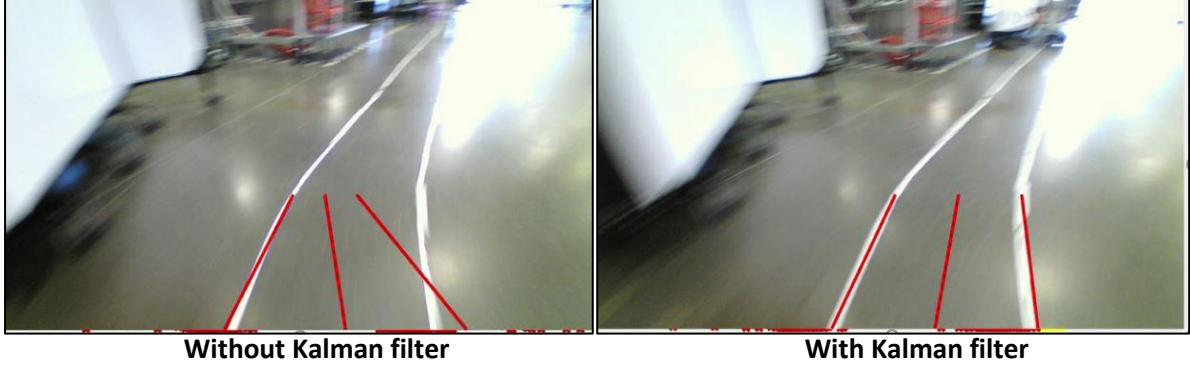


Figure 9: Stabilization of lines due to Kalman filter

2.1.7. Inverse Projection

Inverse projection is done in order to project the image coordinates to vehicle coordinates. This makes vehicle control more intuitive and it is easy to deal with vehicle movements in vehicle reference frame. We did this conversion in two stages – first, the images coordinates were converted to world coordinates and then the world coordinates were converted the vehicle coordinates. The first conversion was done using the inbuilt MATLAB function ‘pointsToWorld’.

$$Worldframe[x, y] = pointsToWorld(cameraParams, rotationMatrix, translationVector, imageframe[x, y])$$

So basically, this function converted the line coordinates obtained in image frame to world frame and then adjustments were done by physical measurements in X and Z direction to transform the world coordinates to vehicle coordinates.

2.1.8. Lane Tracking Controller

Stanley controller was designed for the vehicle’s lane tracking control due to its accuracy and ease of application. This controller uses Stanley method to find the steering angle required to negotiate turns. The Stanley controller gets two inputs – angle of departure from the desired path and distance of the closest point of the desired path. Based on these two inputs, the Stanley controller determines the steering angle required to follow the desired path. The departure angle (θ_e) and distance of the closest point to the desired trajectory (e_{fa}) is calculated as follows –

$$\theta_e = -\arctan \frac{(Z_{cf} - Z_{cn})}{(X_{cf} - X_{cn})} - 90$$

$$e_{fa} = X_{cn} + \frac{(X_{cf} - X_{cn})}{(Z_{cf} - Z_{cn})} \cdot (Z_{cn} - C_{pos})$$

In its simplified version, the steering angle of the vehicle (φ) is calculated as follows –

$$\varphi = -K_1 \cdot \theta_e - K_2 \cdot e_{fa}$$

Here, K_1 and K_2 are tuning parameters.

The steering angle obtained from the controller was sent to the Arduino which controlled the steering servo and helped maintain the desired vehicle trajectory.

2.2. Experimental Results

This section of the report summarizes the experimental data obtained from lane detection and discusses the data obtained from lane tracking control. Subsequently, the results would be discussed in the same order as that of the technical approach.

2.2.1. Summary of Lane Detection Results

Edge detection

Some of the iterations done with the canny algorithm for edge detection are explained in technical approach section. Figure (10) is the iteration we finalized for edge detection. The canny parameters for this iteration are [0.2, 0.36].



Figure 10: Edge detection using canny algorithm

Region of interest polygon

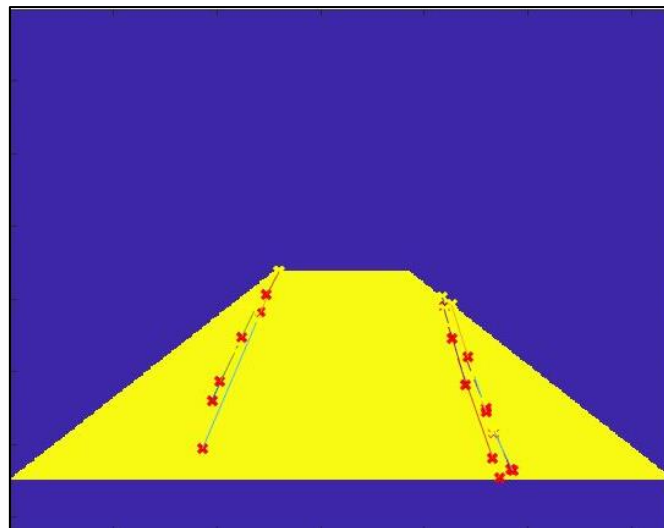


Figure 11: Region of interest polygon

Hough transform

Figure (12) shows finalized Hough space lane detection with number of houghpeaks = 8.

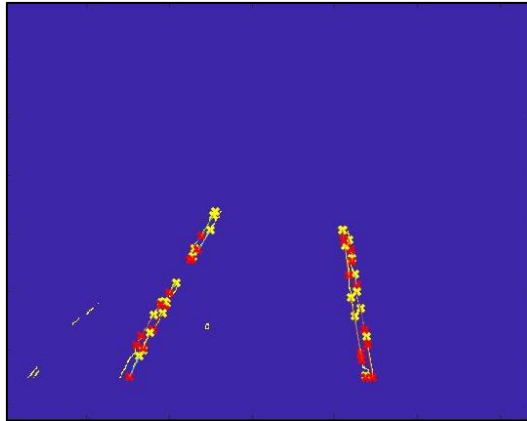


Figure 12: Hough transform

Post-processing

Kalman filtering used to stabilize the detected lines.

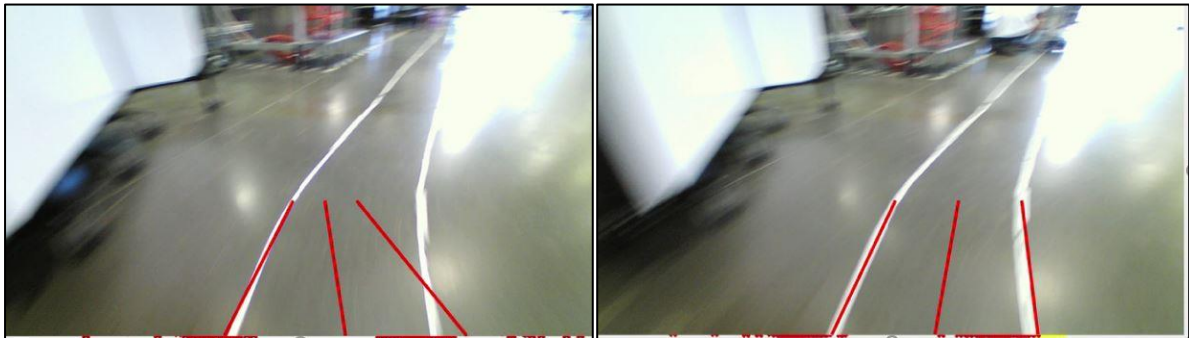


Figure 13: Stabilization of lines due to Kalman filter

2.2.2. Experimental Results of Lane Tracking Control

The values given to the steering servo from lock-to-lock steering wheel conditions were 0-1, where anything less than 0.5 gives a left turn and anything above 0.5 gives a right turn and 0.5 being the neutral state of steering wheel. But as per the test track turns, utilizing full capacity of steering angle was not needed. Hence, we limited the steering angle which we get as an output from Stanley controller between 0.25 and 0.75. The method by which it was done in MATLAB is shown below.

```
if phi >= 0.75
    phi = 0.75;
elseif phi <= 0.25
    phi = 0.25;
else
    phi = phi;
end
```

Figure 14: Limit steering angle

Also, we faced this issue quite a few times when one of the two lines went undetected and the vehicle would lose control over lane tracking. This situation occurred when the code was just able to extract lines from one side in the Hough space. This issue is shown in the figure below.

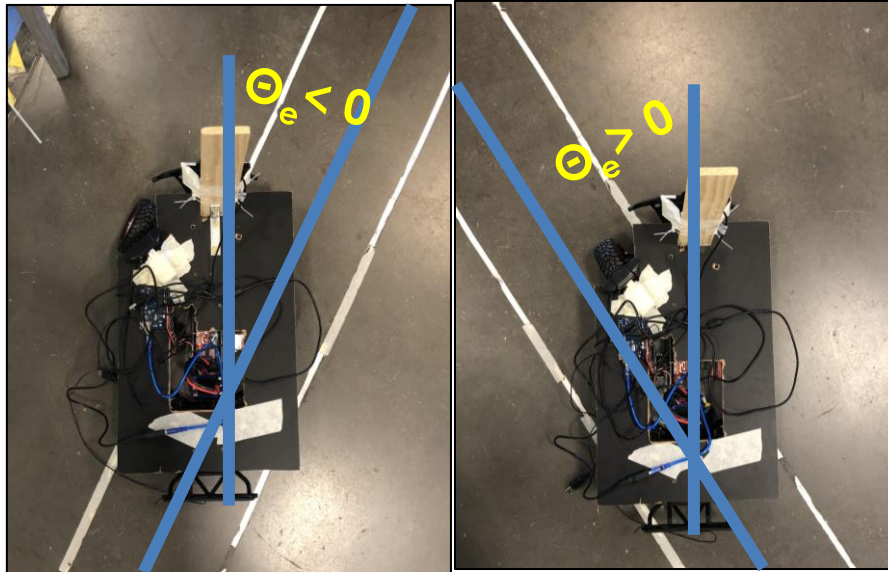


Figure 15: Missing right line

This issue was addressed by telling the vehicle what to do under these circumstances. So basically, instructions were given for four exhaustive situations here –

- When θ_e is greater than zero and left lines go undetected – take left turn
- When θ_e is less than zero and left lines go undetected then – take right turn
- When θ_e is greater than zero and right lines go undetected – take left turn
- When θ_e is less than zero and right lines go undetected – take right turn

Figure (16) helps understand these conditions pictorially.



```

if isempty(leftlines) && theta_e>0
    phi = 0.3;
elseif isempty(leftlines) && theta_e<0
    phi=0.6;
else
    phi=phi;
end

```

```

if isempty(rightlines) && theta_e>0
    phi=0.3;
elseif isempty(rightlines) && theta_e<0
    phi=0.6;
else
    phi=phi;
end

```

Figure 16: Solution to missing line problem

The results for lane tracking control shows the variation in lane departure angle (θ_e), distance of closest point on the desired path (e_{fa}) and road wheel steering angle (ϕ) with respect to time samples for one whole lap on the test circuit.

Variation in e_{fa} for one lap on the test track

Figure (17) shows the variation in e_{fa} when the vehicle goes around the test track for one lap. The vehicle started at the straight patch of the track with vehicle aligned to it, hence, initially the e_{fa} is almost close to zero. As the vehicle goes around the turns on the track, there are slight oscillations in the values of e_{fa} about zero which means whenever the vehicle goes away from its desired path, the controller tries to bring it back to the desired path but due to steering servo lag or vehicle steering system lag, the vehicle corrects its path more than what is needed and this causes oscillations in the trajectory of vehicle.

Some of the abrupt peaks in this graph are due to the conditions when either left lines or right lines are not detected and hence, at those points the value of e_{fa} makes no sense.

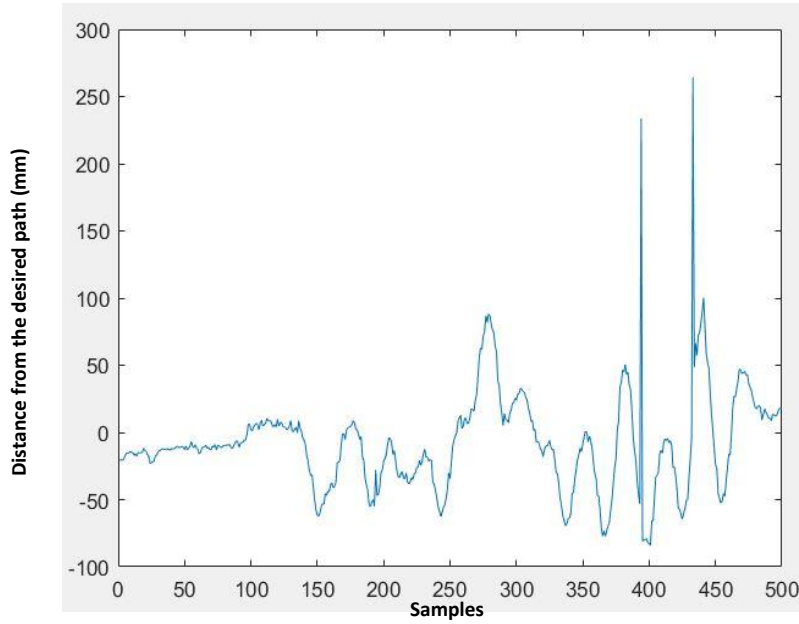


Figure 17: Value of e_{fo} for one lap on the test track

Variation in θ_e for one lap on the test track

The variation in θ_e shows almost the same trend as e_{fa} , initially the departure angle is nearly zero since, the test was started from straight patch of the track with vehicle aligned to it. As the vehicle maneuvers through the turns on the test track, there is a variation in departure angles. The two spikes again in this graph are due to the same reason as that in the e_{fa} graph. These are due to undetected left or right lines.

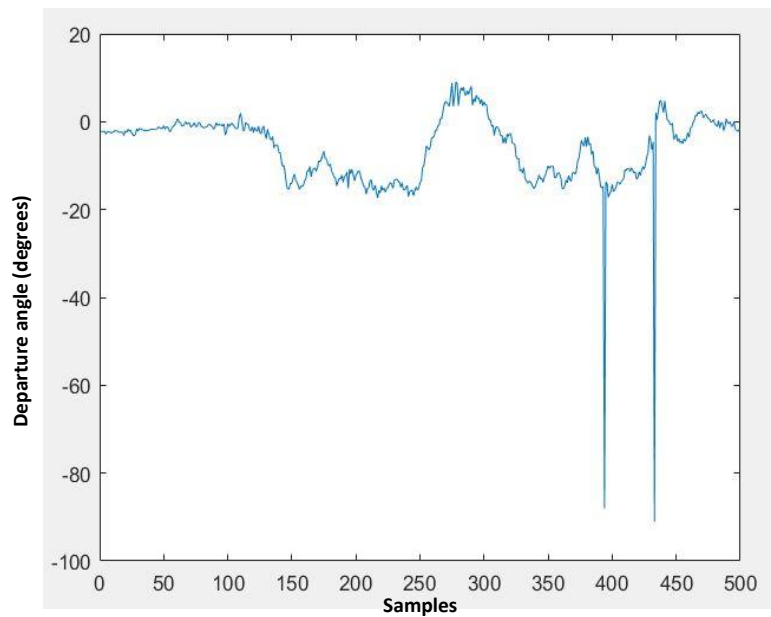


Figure 18: Variation in departure angle for one lap on the test track

Steering angles given by the controller to the steering servo

As per the change in θ_e and e_{fa} , the controller generates a value for steering angle so as to make the vehicle track the desired path all the time. At 0.5, the steering is in its neutral state, thus when we began our vehicle's lap, initially on the straight path, the controller made small changes to the steering angles around the value 0.5 so as to bring the vehicle to its desired trajectory. After that, the controller generates a number between 0.25 and 0.75 depending upon the error in θ_e and e_{fa} .

At the points where we saw undesired spikes in the θ_e and e_{fa} graphs, steering angle at those points is still under control and hence the vehicle is able to track the lane or return back to the desired lane at all times.

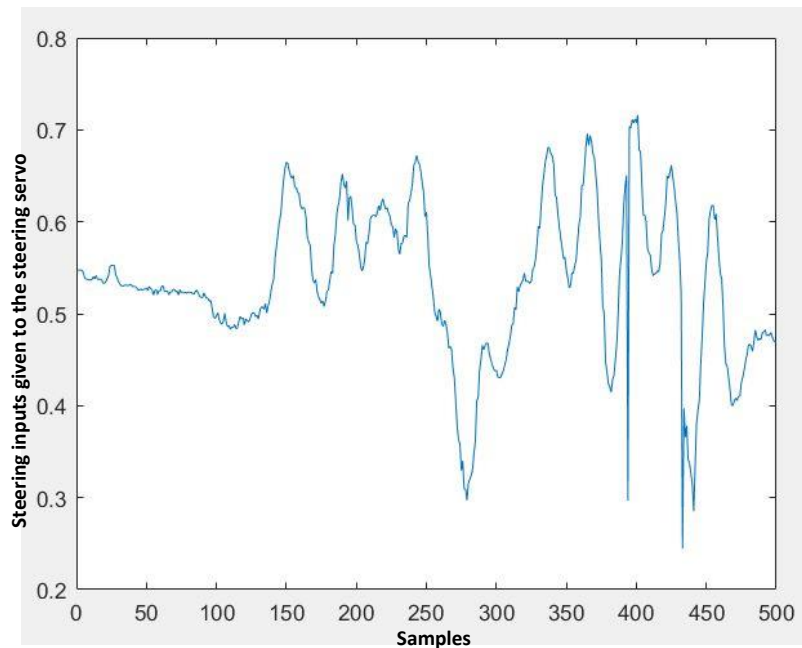


Figure 19: Steering input to the steering servo from the controller

3. Road Sign Recognition and Vehicle Reaction

3.1. Technical Approach

Process Flow

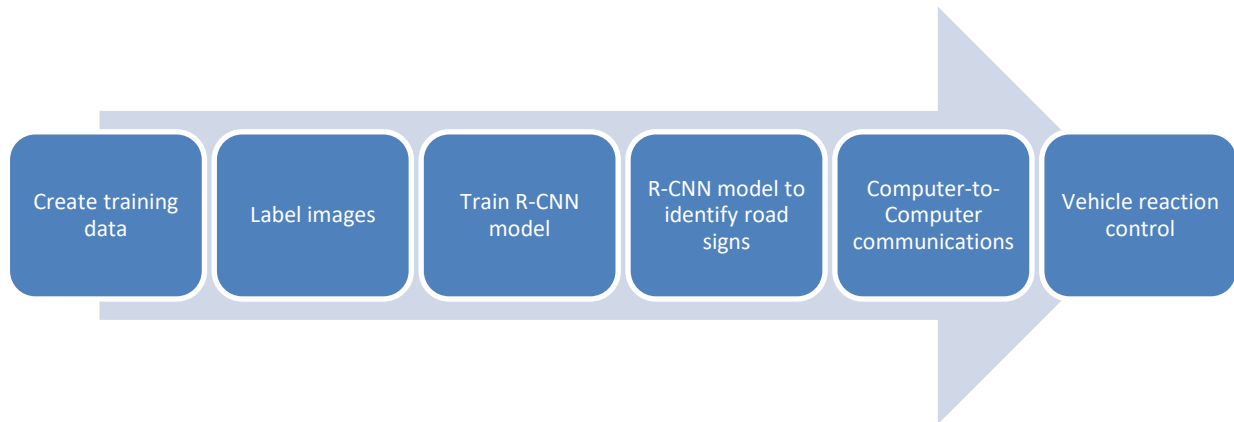


Figure 20: Process flow for road sign recognition and vehicle reaction

3.1.1. Create training data

Task was to recognize ‘Stop’ and ‘School’ sign. This was achieved using deep learning approach. For training the neural network, we began with taking as much photos, of the two sign boards, as possible and from all possible angles. Thus, we used almost 150 images of each sign board to train the network.

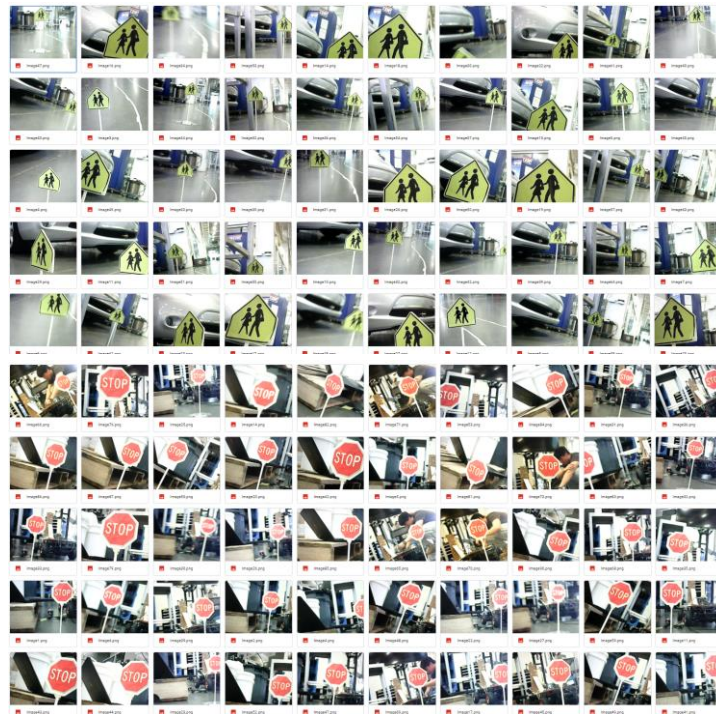


Figure 21: Training images of the two road signs

3.1.2. Label Images

Image labelling was done using the MATLAB deep learning toolbox. The procedure is quite simple for this, all the training data is loaded on the MATLAB deep learning app and bounding boxes are defined in each photo around the road signs that are to be recognized. The bounding box is labelled as per the

road sign it contains in it. There were only two road signs in the scope of this project – stop sign and school sign, hence, two labels were defined as ‘Stop’ and ‘School’ to segregate all the training images.

3.1.3. Train R-CNN Model

Region-based Convolutional Neural Network (R-CNN) was used to perform the road sign recognition. R-CNN model is used due to its relatively simple training procedure and good reliability. It is also faster than the CNN Transfer learning since it uses cropped and small region images. R-CNN network was trained using CIFAR10 Network.

3.1.4. Trained R-CNN Model to Identify Road Signs

Trained R-CNN model was then tested to check the accuracy of its sign recognition and we faced no problems on this part of the project since, the R-CNN model was trained with large number of photos and from all the angles possible. So almost every time the side mounted camera was able to recognize the road signs. Figure below just shows a sample of road sign recognized by our model.



Figure 22: Road sign detection

Once the road sign is recognized, we wrote a code in MATLAB to quantify the road sign recognition result. To put it in simple words, a code was written to assign variable ‘data’ a value 1 when a stop sign was recognized and a value of 0 when school sign was recognized. Cases when there are no road signs in the sight of the camera, the code will assign value 2 to variable ‘data’. The MATLAB code for the same is shown in figure (23).

```
if label(idx) == 'stop'
    data=1;
elseif label(idx) == 'school'
    data=0;
else
    data=2;
end
```

Figure 23: MATLAB code to quantify road sign recognition

3.1.5. Computer to Computer Communications

Two computers were used to receive the sensory data in this project. The first computer was used to receive sensory data from front camera which was used for lane tracking. Second computer was used to receive sensory data from the side mounted camera which was used for road sign recognition.

Hence, to control the vehicle using sensory data from both the sensors, it was necessary to establish a medium for communication between two computers.

There were two options to establish a communication network - User Datagram Protocol (UDP) or Transmission Control Protocol (TCP). UDP was used for communications between the computers. Although there are a lot of down sides to using UDP as it is not reliable, possibility of loss of datagrams, no sequencing, etc., it has one inherent benefit that it is a lightweight protocol. Since, our project is not really affected by reliability or loss of data, and we want small-size data transfer at fast rate, UDP setup best suits this project. Thus, a UDP connection was established between the two computers for sending sensory data of road sign detection from second computer to the first computer.

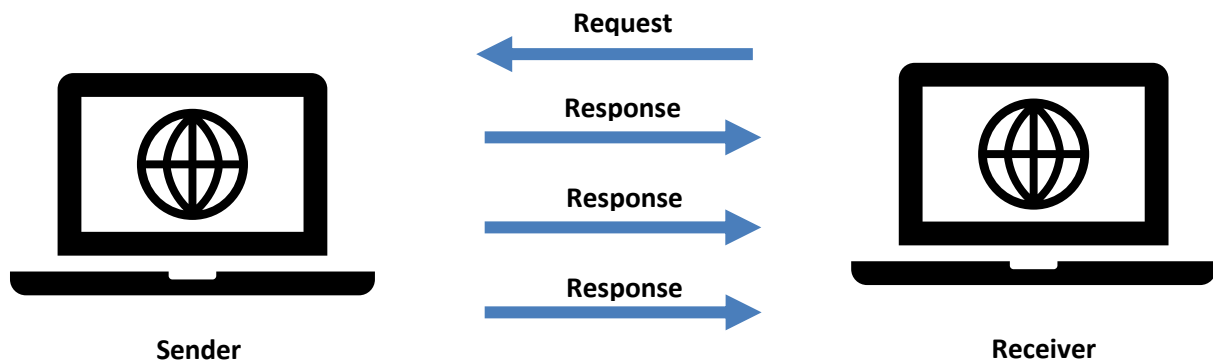


Figure 24: UDP communication layout

3.1.6. Vehicle Reaction Control

After recognition of the road signs, the last task of the project was to control the vehicle such that when it sees the school sign, it reduces its speed to half until it sees the stop sign. At stop sign, the vehicle must stop for two seconds and then run at a high speed until again it sees the school sign. To achieve this, the 'data' variable generated in second laptop should send the data to first laptop using UDP communication and then desired controls need to be applied through the code in first laptop.

First, the performance control requirement of the vehicle was visualized by dividing the track map in three regions – before school sign, after school sign and at stop sign. These three divisions are shown in the track map below.

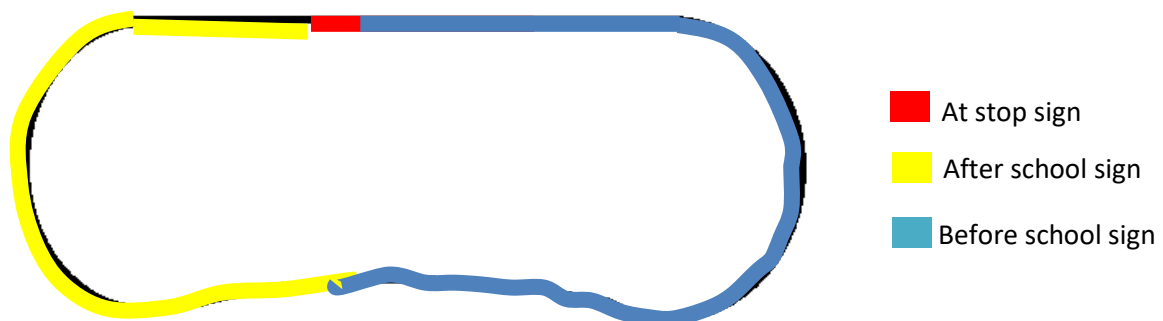


Figure 25: Division of map of test track

As we already discussed, a variable 'data' was created for quantifying the road sign identification, similarly, a variable 'state' was defined to make the vehicle behave differently in the after-school

region and before-school region since in both the regions, the second computer will send value of data as 2 and therefore, it will be impossible to have different vehicle speed in these two regions. So, with the help of the two variable – data and state, we achieved the goal of regulating the vehicle speed in the blue and yellow regions.

The figure below shows the MATLAB code for achieving this speed control. To put the conditions in

```

if data == 0
    state = 0;
elseif data == 1
    state = 1;
end
if state == 0
    writePosition(v,0.55)
    pause(0.1)

    writePosition(v,0.49)
    pause(0.15)

elseif state == 1 && data == 1
    writePosition(v,0.5)
    pause(2)

    writePosition(v,0.6)
    pause(0.35)
    writePosition(v,0.49)
    pause(0.15)

elseif state == 1 && data == 2
    writePosition(v,0.56)
    pause(0.12)

    writePosition(v,0.49)
    pause(0.15)
end

```

Figure 26: Vehicle Reaction Control

simple words, the code says that when the side camera detects a school sign, i.e. when data = 0, keep state of vehicle equal to zero and when the side camera detects a stop sign, i.e. when data = 1, keep the state of the vehicle as 1. Now, we impose some conditions on the combination of state and data values to have vehicle reaction control. So, if the state of the vehicle is equal to zero, i.e. the vehicle is in the yellow region, so the speed of the vehicle should be reduced to half. Next condition is if values of both state and data are 1, then the vehicle is at the stop sign (red region) and hence, it will pause for two seconds and then run at a high speed. The last condition is if the state of the vehicle is 1 and value of data is 2, then the vehicle is in the blue region and thus it has to run at a high speed. These conditions can be understood better with the help of track diagram below.

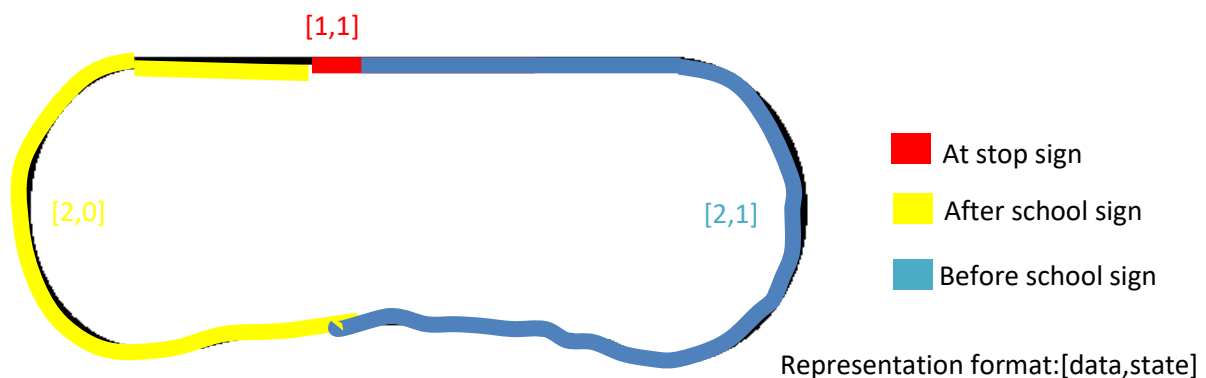


Figure 27: Vehicle reaction control explained

3.2. Experimental Results

3.2.1. Road sign recognition, Communication and Vehicle Reaction Results

The graphs for variables 'state' and 'data' are plotted out to explain the road recognition, computer-computer communication and vehicle reaction results.

Plot for variable 'data' sent from second computer to first computer

As discussed in the technical approach section, a variable 'data' was defined to quantify which road sign was recognized, stop sign or school sign. Thus, a value of 1 was given to variable data when stop sign was recognized and value 0 was given when school sign was recognized. When there was no sign in the image frame, the value of data was set to 2. From the figure shown below which plots the value of data for one lap of the test track shows the detection of school and stop sign. The lap began just before the stop sign hence, the value of data is 2. When the side camera detects the stop sign, the value of data changes to 1 and then again goes back to 2 when no road sign is detected in the image. After some time samples the value of data falls to zero which indicates that the side camera has recognized school sign. Almost at the end of the lap the vehicle again detects the stop sign and gives a value 1 to data

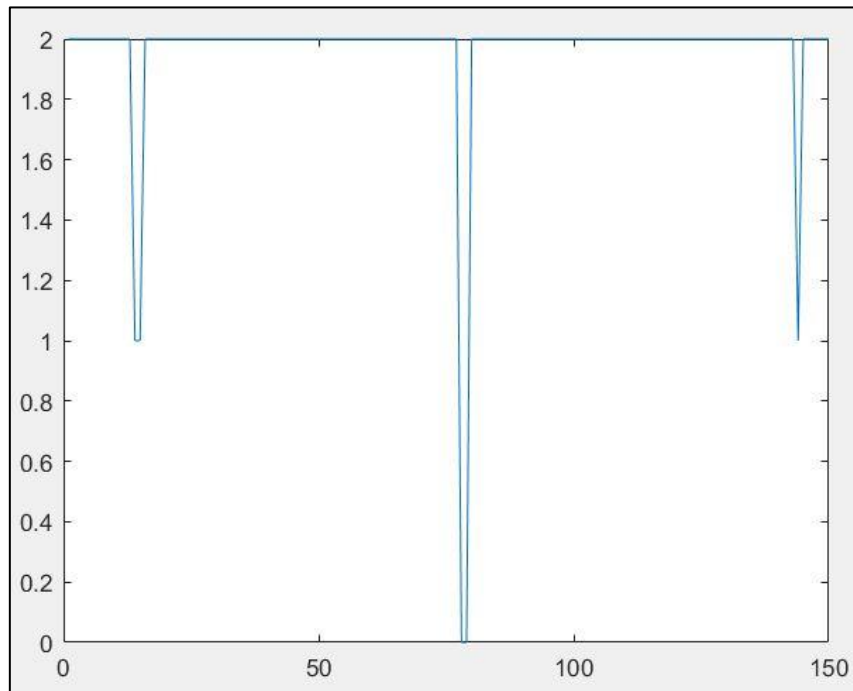


Figure 28: Varying value of data with time samples

Plot for variable 'state' decided by the first computer based on the 'data' value received from the second computer

We gave two states to the vehicle – state = 1 and state = 0. It was defined that when data = 0, state = 0 and when data = 1, state = 1. Thus, when the vehicle starts from stop sign, the data is 1 and hence, state of the vehicle is 1. It maintains its state till the school sign is recognized. After recognizing the school sign, the state of the vehicle changes to zero.

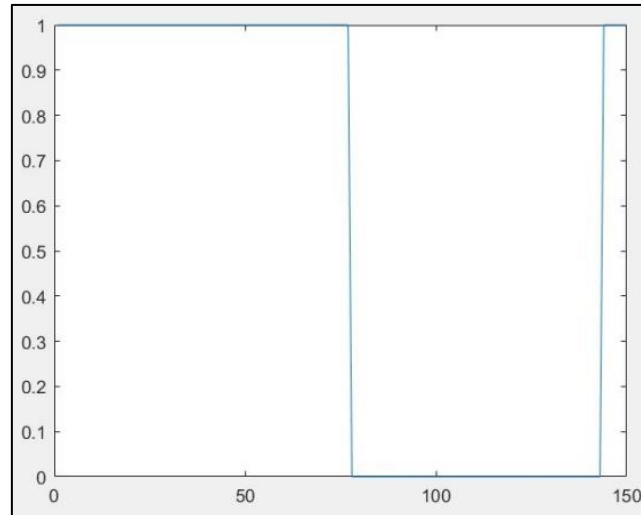


Figure 29: Varying value of 'state'

Vehicle Reaction Controls

The vehicle reaction control is explained in detail in the technical approach section. Basically, depending on the combination of state and data values, the reaction control of the vehicle is regulated. The figure below shows the three different regions the test track is divided into based on difference in vehicle reactions in these regions, combination of data and state value that decided vehicle reaction and speed of the vehicle in these regions.

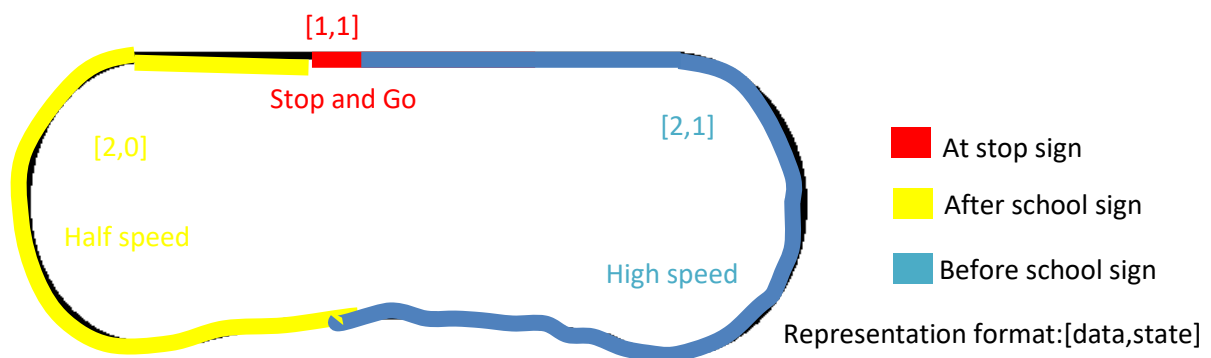


Figure 30: Vehicle reaction control

4. Project Achievements

- Got acquainted with computer vision toolbox of MATLAB and understood how to practically apply the computer vision concepts for lane tracking.
- Learnt transformations from image frame-world frame and vice-versa.
- Application of Kalman filter.
- Road sign recognition was done with deep learning concepts and our team got it done in first attempt.
- Learnt basics of deep learning and understood practical implementation of R-CNN.
- Designing Stanley controller is I feel a great achievement and a key take away from this project.

5. Project Challenges

- To find optimal height for mounting lane tracking camera was one of the initial challenges.
- Inverse projection was kind of a tricky part, we took a lot of time to figure out how to and why to convert the image coordinates to vehicle frame.
- Unstable left and right lines in lane detection posed a challenge for us until we implemented a Kalman filter.
- We took some time to understand Hough peaks and Hough lines.

References

1. *Canny Edge Detector.*, 2019. Web. May 2, 2019.

Course Evaluation

Student Assessment of Instructors		Logout
<p>Welcome Nayan Deshmukh to the Online Instructor Evaluation System. The following evaluations have been set up for you at this time. Their scheduled availability is indicated.</p> <p>Choose any evaluation that is currently available to complete:</p> <p>Thank you for helping improve instruction at Clemson University. <i>Your evaluation of AUE 8930 401, Spring Semester 2019 - Yunyi Jia has been completed and your connection to the answers has been removed.</i></p> <ul style="list-style-type: none"> <input type="radio"/> AUE 8500 006 - Beshah Ayalew - Completed. Responses will be available to the instructor after grades are posted at the end of the semester. <input type="radio"/> AUE 8820 010 - Srikanth Pilla - Not available now. Scheduled to be available, beginning 4/5/2019 6:00 AM and ending 4/26/2019 11:30 PM, from 6:00 AM to 11:30 PM each day. Responses will be available to the instructor after grades are posted at the end of the semester. <input type="radio"/> AUE 8820 010 - David William Schmueser - Not available now. Scheduled to be available, beginning 4/5/2019 7:00 AM and ending 4/26/2019 11:30 PM, from 7:00 AM to 11:30 PM each day. Responses will be available to the instructor after grades are posted at the end of the semester. <input type="radio"/> AUE 8820 010 - Christiaan Jos Jan Paredis - Not available now. Scheduled to be available, beginning 4/5/2019 7:00 AM and ending 4/26/2019 11:30 PM, from 7:00 AM to 11:30 PM each day. Responses will be available to the instructor after grades are posted at the end of the semester. <input type="radio"/> AUE 8821 200 - Srikanth Pilla - Not available now. Scheduled to be available, beginning 4/5/2019 6:00 AM and ending 4/26/2019 11:30 PM, from 6:00 AM to 11:30 PM each day. Responses will be available to the instructor after grades are posted at the end of the semester. <input type="radio"/> AUE 8821 200 - David William Schmueser - Not available now. Scheduled to be available, beginning 4/5/2019 7:00 AM and ending 4/26/2019 11:30 PM, from 7:00 AM to 11:30 PM each day. Responses will be available to the instructor after grades are posted at the end of the semester. <input type="radio"/> AUE 8821 200 - Christiaan Jos Jan Paredis - Not available now. Scheduled to be available, beginning 4/5/2019 7:00 AM and ending 4/26/2019 11:30 PM, from 7:00 AM to 11:30 PM each day. Responses will be available to the instructor after grades are posted at the end of the semester. <input type="radio"/> AUE 8930 003 - Imtiaz UL Haque - Not available now. Scheduled to be available, beginning 4/5/2019 6:00 AM and ending 4/26/2019 11:30 PM, from 6:00 AM to 11:30 PM each day. Responses will be available to the instructor after grades are posted at the end of the semester. <input type="radio"/> AUE 8930 401 - Yunyi Jia - Completed. Responses will be available to the instructor after grades are posted at the end of the semester. <p>The evaluation of your courses and instructors is designed to be anonymous. However, the process for completing your evaluations requires that a temporary connection be maintained between your answers and your identity. This connection is severed upon completion of each evaluation. If you cannot complete an evaluation in one session, you may return to finish it later. If your evaluation is interrupted due to computer or network problems, all answers previously recorded will be preserved and you may return to complete the evaluation when the problems have been resolved. In the event that you have unfinished evaluations when the evaluations are turned off at the end of the semester, the connections to your identity will be removed automatically before the results are released to the instructor.</p> <p>If you have problems or wish to make constructive suggestions for improvement, please send a message to CCTT Help or call 656-0161.</p>		