Advanced Lane Keeping Assist

Emma Ní Bhriain
Dept. Science and Computing
Waterford Institute of Technology
Waterford, Ireland
obrienemma0@gmail.com

***Abstract*— This report documents the design and implementation of Advanced Lane Keeping Assist as an ADAS feature. This feature is used on roads without road markings in order to control the position of an autonomous vehicle. This feature was implemented using Matlab. For the purpose of this assignment, the feature was demonstrated using an image of a road without markings.**

I. Introduction

Advanced Lane Keeping Assist is used for roads without road markings. The width of the road is considered in this analysis and the area that the lane should cover is then calculated. The ideal positioning of the vehicle is then determined from this calculation. When the vehicle drives on a road on which there are road markings, the feature becomes inactive. The system uses online maps to recognise the type of road that the vehicle is on, such as whether the road is a one-way road or not. In the event of a narrow road, if an oncoming vehicle is detected, the system re-adjusts the lane dimensions accordingly and may signal to the driver to pull in and give way to the oncoming vehicle. In a more advanced version of this system V2V communication could be explored as an option when determining which vehicle should yield in this situation.

II. Rational, Goals and Benchmarks

A. Rational

The rational is to guide the autonomous vehicle into its ideal positioning based on the width of the road.

B. Goals

The goal of the system is to calculate a vehicle's ideal position on a road with no road markings.

C. Benchmarks

A number of benchmarks can be used to assess the success of this feature:

- The system must be able to identify the edge of the road.
- The system must be able to divide the width of the road in half.
- The system must indicate to the driver where the road markings should be.
- Highlight the lane in which the car should be driving

III. Design

A. Sensor Data

This feature takes data from external sensors such as cameras to analyse the road ahead. This data is combined with data from the cloud which indicates the type of road that the car is on, for example if the car is on a one-way road it does not need to calculate the position of two lanes as there will be no on-coming cars. In the event that the car moves onto a road where there are road markings visible, this system will turn off and another lane keeping assist algorithm can take over.

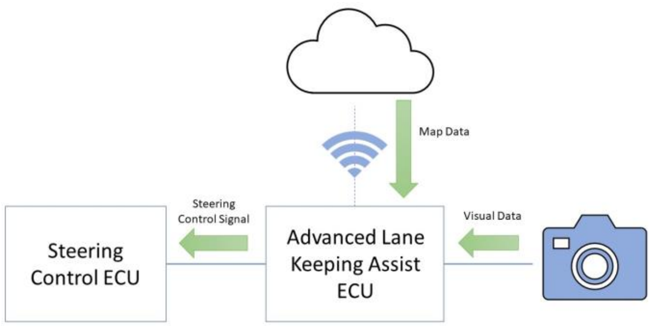
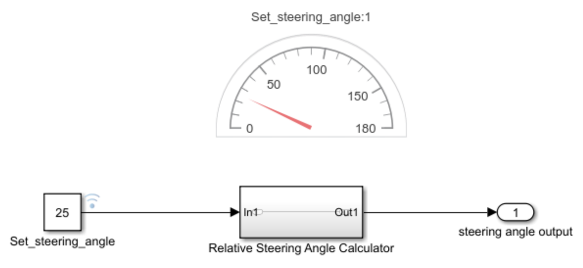


Fig. 1. System Comunication Design

B. Matlab/Simulink Design

The Advanced Lane Keeping Assist feature controls the steering of the vehicle. In order to visualise this in Simulink, I created a simple model in which the variable 'set_steering_angle' is set by the lane_keeping.m script. The Simulink script is named simple_steering.slx and is opened automatically when the lane_keeping.m script is run.

For the purpose of the demonstration, the steering angle is set to 25. The range of angles is 0 to 180, 90 degrees is when the wheels are positioned straight ahead. 0-89 is when the wheels are pointed to the left and 91-180 is when the wheels are pointing to the right. The steering angle is displayed in a 'Half Gauge' block in the Simulink model.



IV. IMPLEMENTATION

This project was implemented using Matlab and Simulink. The main algorithm used in the Matlab section of the project was the Hough transform algorithm.

A plant model of a car was used in Simulink to simulate the communication between the Matlab algorithm and the steering control.

This feature detects the edge of the road and measures the distance between each edge. This distance is then divided by 2 and a line is drawn along the center-line of the road creating two lanes. Map data is used to make a decision on which lane the car should drive in as well as confirming the type of road, i.e. a one-way road. Messages are then sent to the steering ECU to keep the car within the calculated lane.

V. CALCAULATIONS

A. *Calculating the edge of the road*

The Hough Lines were used to draw the edge of the road onto the image of the road. The equations of these lines were calculated using the formula

$$y = mx + c \qquad (1)$$

The slope m was calculated using the formula

$$m = \frac{y2 - y1}{x2 - x1} \qquad (2)$$

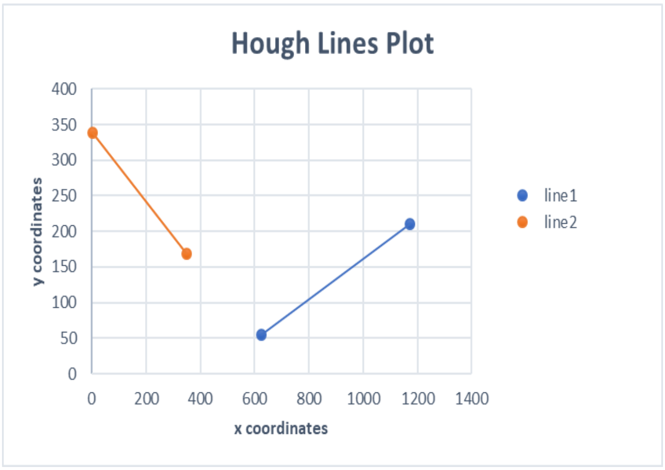
The start and end points of the Hough lines were used as the coordinates for calculating the slope of each line. For line 1 in the sample image road.jpg, the starting coordinate was (626, 55) and the end-point was (1171, 211) giving a slope of 0.2862 for this line.

$$m = \frac{211 - 55}{1171 - 626} = -0.2862$$

Similarly, for line 2, the starting coordinates are (3, 339) and the ending coordinates are
(351, 169). The slope for this line is therefore -0.4885.

$$m = \frac{169 - 339}{351 - 3} = 0.4885$$

These lines were plotted on a 2-dimensional plane to allow for easier visualisation.



B. *Calculating the Road Lanes*

In order to split the road into two lanes, a new line needed to be calculated based on the Hough lines detected in the image. This line would be positioned midway between the two Hough lines so as to split the road into two equal parts. A point along line1 was chosen and the point at which line 2 had the same value for its y-coordinate was calculated. The x-coordinates of the two lines at this y value were then added and divided by 2 to calculate a midpoint. This process was repeated, and these points were then used to calculate the equation of the newly calculated line and then display this line on the image.
The formula for the midpoint (*M*) between two coordinates is

$$M = (\frac{x1 + x2}{2}, \frac{y1 + y2}{2}) \qquad (3)$$

The first points to be used to calculate a midpoint were the point on each of the lines when y = 0. By manipulating the equation of a line as seen in (1), the corresponding x-value can be calculated using the formula (4) below.

$$x = \frac{y - c}{m} \qquad (4)$$

When y = 0, the x-value on line 1 is 433.8526, giving a co-ordinate of (433.8526, 0).

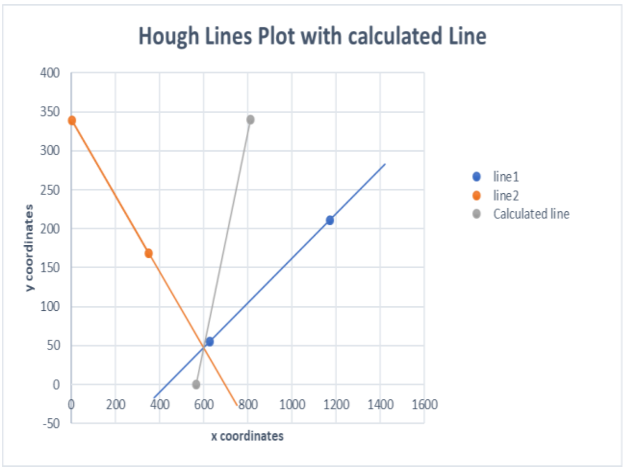
When y = 0, the x-value on line 2 is 696.9529, giving a co-ordinate of (696.9529, 0).

The midpoint between these two coordinates is (565.4028, 0).

$$M = \left(\frac{433.8526 + 696.9529}{2}, \frac{0 + 0}{2}\right) = (565.4028, 0)$$

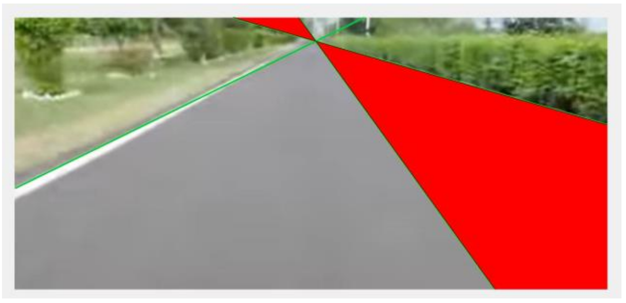
This process was repeated using the point (1, 339.4885) along line2. The x-value on line1 when y = 339.4885 is 995.5928. This gave the new midpoint
(811.2964, 339.4885).

Using these two new coordinates (565.4028, 0) (811.2964, 339.4885), the equation of the line was calculated, and the line was plotted onto the image of the road using formulae (1) and (2).



C. *Highlighting the correct side of the road*

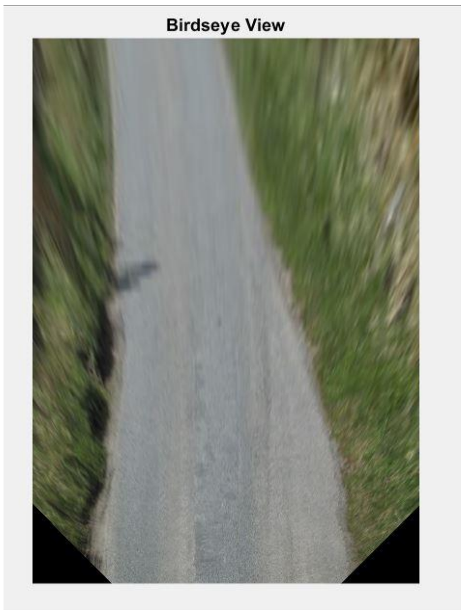
Depending on the location of the driver, either the left-hand lane or the right-hand line is highlighted to indicate which lane the car should be positioned in. This was done by using the patch function in Matlab. The area between the central line that created a lane and the edge of the road was filled with the colour red to indicate the lane in which the car should drive in.



VI. Alternative Approach

A. *Birdseye view*

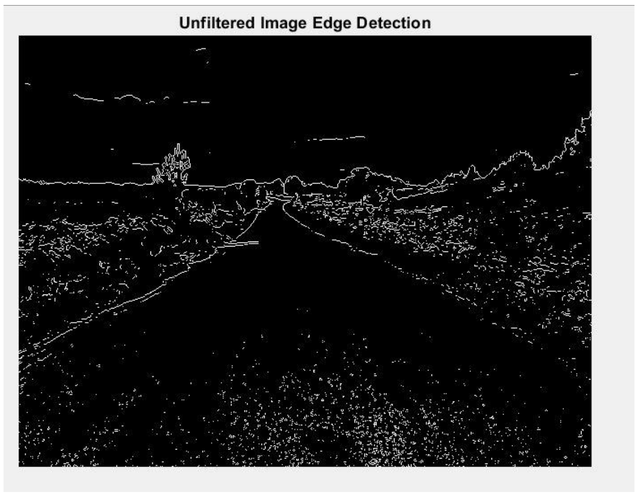
Whilst investigating ways in which the program could work, I attempted to convert the image of the road into a birds-eye view using the transformImage() function in Matlab. An example of this attempt can be found within the file *birdseye.m*.
Before alteration, the image showed a head on view of the road. After running the script *birdseye.m*, the same road is visualised from above.



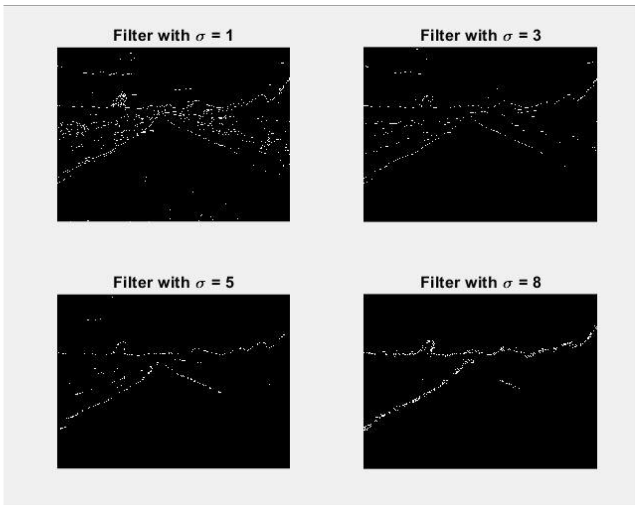
B. *Filtering Image*

When trying to identify the edges of the road, I experimented with a number of different edge detection algorithms including Sobel, Canny, Roberts. I used these in conjunction with some Gaussian smoothing filters to eliminate noise that appeared due to a natural pattern in the road surface.

As can be seen here, there is a lot of noise in the foreground of the image as the tarmac is not completely smooth.



The imgaussfilt() function in Matlab allowed me to applying a smoothing filter to the image to remove noise and allow for more accurate detection of the edge of the road. This filtering was carried out using various standard deviation values which resulted in various levels of filtering.



VII. Results and Analysis

A. Benchmarks

The system can identify the edges of the road using the Sobel edge detection algorithm. In the image road.jpg, the edge of the right hand side of the road is detected where there is a bollard, rather than where the road ends. This could potentially be solved by restricting the region of interest of the image to crop out the area of the image that is not road.
Dividing the width of the road was achieved by using geometry to calculate a line along the midpoints of the two lines representing the edge of the road.

The system draws Hough lines onto the image in order to indicate where the road markings should ideally be placed.

The lane in which the driver should be driving is highlighted in red. This works well for when the driver should be driving on the right hand side of the road. I encountered some issues when highlighting the left lane however as the area between the centre line and the left edge of the road was highlighted on the sides rather than along the road. This could also possibly be fixed by altering the co-ordinates in the patch() method to the region of interest in the image which is the road and not the surrounding area.

B. Calculating the steering angle

In this demonstration, the setting of the steering angle in the Simulink model was hard-coded to show how the communication between the lane assist system and the vehicle controls would work. In a working version of this system, the steering angle would be calculated by reading the position of the vehicle in relation to the centre of the lane and readjusting accordingly.

VIII. Further Development

As I displayed this feature on an image of a road, the next step would be to implement the same feature onto some video footage of a road which would involve continuous calculations.

This feature should also detect that a road does have markings in order to know to switch off the lane calculation program.

In a future development of this feature, the system could take the width of the vehicle into account when choosing the positioning within the lane.

An extension of this system could also detect oncoming vehicles on a narrow road and determine which vehicle should have right of way based on whether there is enough room for both vehicles to pass or based on which vehicle has room to pull in and allow for the other to pass. This would require Vehicle-to-Vehicle communication (V2V).

References & Sources

[1] Mathworks, ‘transformImage()’, Available at https://uk.mathworks.com/help/driving/ref/birdseyeview.transformimage.html#bvh354t-1-birdsEye

[2] F. Walsh, ‘lab-hough-transforms’, Available at https://wit-computing.github.io/adas-2018/topic-1/book-a/index.html

[3] Image: ‘unmarked_road_1’, Available at geograph.org.uk

[4] Image ‘road.jpg’, Frame taken from video ‘Test road for edge detection’, H.Rao, Available at https://www.youtube.com/watch?v=KPR8zR445iA