Emergency Brake Assist System

Emmanuel Sedicol
Department of Maths and Computing
Waterford Institute of Technology
Waterford, Ireland
20072377@mail.wit.ie

***Abstract*— this paper outlines the design and implementation of an Emergency Brake Assist System (EBAS). I will be investigating the process of detecting the distance of a vehicle from the car's front bumper using the concepts of lane detection and image processing. Based on the calculated distance, the EBAS will either takeover the braking system of the car or not.**

Keywords – ADAS, Matlab, Hough, distance, brake system

I. Introduction

In the automotive industry one of the growing causes for accidents is when drivers accidentally press the gas pedal instead of the brake pedal. With the rapid advance of machine learning and computer vision algorithms EBAS is able provide a very accurate estimation of the distance between your front bumper and the car in front. This paper overlooks the use of image processing in Matlab by extracting frames from a video and executing appropriate algorithms to resolve the defined issue. Overall the main objective of this project is to prevent drivers from crashing the car due to inactive response to danger warnings.

II. Emergency Brake Assist

A. Background

Generally, the EBA system is built with a Radar sensor instead of a camera. Radar sensors work by emitting radio waves around the car and is bounce back to the machine the moment is comes into contact with any object. An intelligent system would then use this concept to try and recreate the surrounding environment.

B. Design

The Emergency Brake Assist will primarily take in a colored static video as its input and extracts video frames inside of Matlab. Each frame will go through a series of image filtering in order to increase the accuracy of the detection algorithm. After the filtering process a line of code is executed to specify the region of interest on the image, this then lessens the size of image to be processed which also lowers the load on the processor. Lastly the system would attempt to detect the lines on the lane and calculate the length of the longest line. The longest line would then be compared to ruler to determine the distance between both the object and vehicle.

C. Input Video

The selected static video which will be used for the development stage of the EBA system is a five second footage of a vehicle driving on a marked lane. The video was taken from an online website [1] because none of the vehicles accessible to me have a camera. Even though it's quite short the video still passes all the requirements needed during the build and testing phase. The only defect about the video is that it contains unnecessary text on both the left bottom and right bottom section.

D. Image Processing

The very first step for the image filtering is to convert the image into grayscale using Matlab's built in grayscale function. The main reason why the image is converted into grayscale first is because the image is required to have two dimensions (rows and columns). Second step is to binarize the frame or converting the image from grayscale to black and white. After binarization I attempted to execute a line detection algorithm, the result showed me there is no more need in adding another filter because the algorithm works on the current filter.

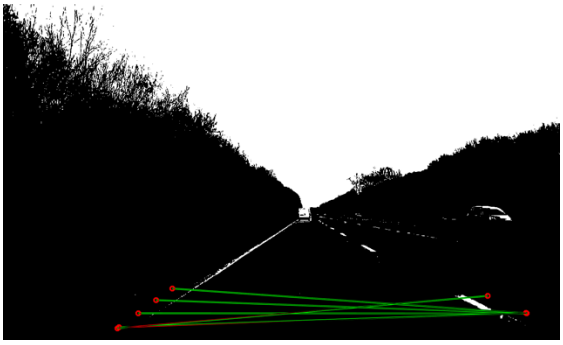
Original Image

rgb2gray Filter Applied

Imbinarize Filter Applied

E. Region of interest

Using the 'ginput' command I was able to select the points of regions that I needed and made the coordinates extraction a little more accurate, all in this a line of code. After extracting the coordinates, I clearly stated using an if and else statement to only allow vertical like edges and prevent edge detection horizontally as show in the figure below.



Horizontal Line Detection Error.

To fully complete the ROI stage the built in Matlab function which takes the frame, row and column coordinates as its parameter is to be executed. This function completely ignores the area outside the defines region, the figure below visually explains the concept.

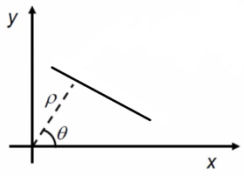
Before ROI Function is Applied

After ROI Function

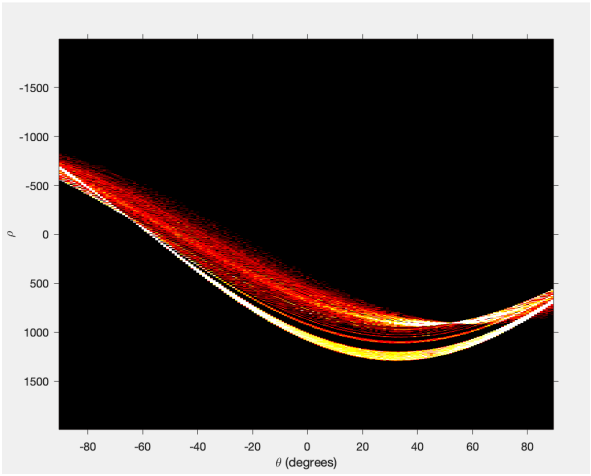
F. Hough Transform

After fully detecting the edges and defining the region of interest we can now apply Hough Transform to detect the lines. Hough Transform uses the following equation to represent a line.

*rho = x*cos(theta) + y*sin(theta)*

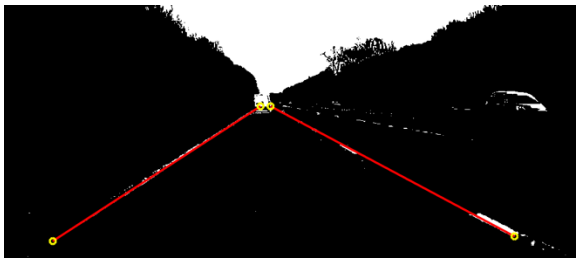


Where *rho* represents the distance from the perpendicular vector to the origin and *theta* representing the angle between the x-axis and the vector [4]. After computing the Hough Transform of the binarize frame we can display the lines detected as sinusoids. The result of the Hough transform can be seen in the figure below.



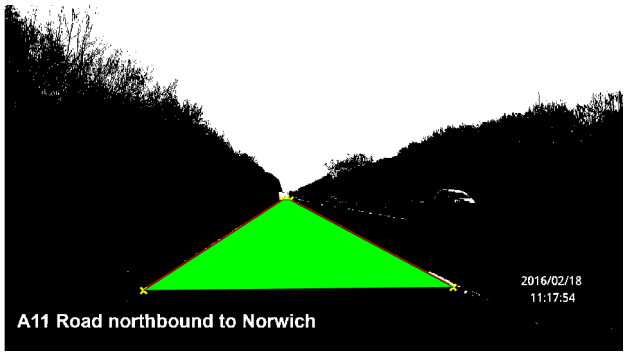
Hough Transform Sinusoids

Using the 'houghpeaks' function we are able to find the peaks in the Hough matrix Once the peak values a located we can implement it on to another function which locates the lines in the image, and we can clearly show this by plotting colored lines. See image below.



Line detection with Hough Transform

Since the end points of the detected lines is available to us, I have decided to put a patch which will visually show the lane in color green.



III. Distance Extraction

A. *Overview*

The main aim of this paper is to investigate definite methods or various solutions to building a working Emergency Brake Assist System in Matlab without the use of any other sensors apart from a camera. The solution I came up with is to get the distance from the vehicle in front to my bumper and compare the distance value to a predefined meter chart as shown in Figure 3.2.

B. *Methodology*

The very first step into completing this solution is to detect the left and right lanes because the total distance will be extracted from either one of those lines. The second step is to write an algorithm that will detect the vehicle in front and determine the line of intersection with the lane markings. To fully complete this step both equation of the lines and slope is needed to calculate the line of interception. Once the line of intercept is calculated, the total distance can be obtained. The total distance will then be compared or matched to the distance meter as shown in Figure 3.2. After the comparison the result will determine the actions of the EBAS. Of course, the EBAS will need connections to multiple ECUs but that subject isn't covered in this paper.

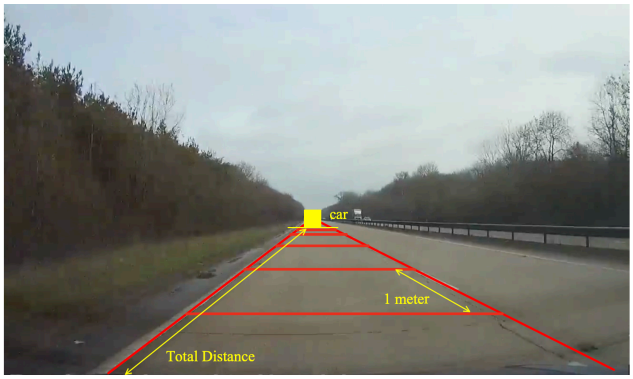


Figure 3.1: *Brief representation of the solution*

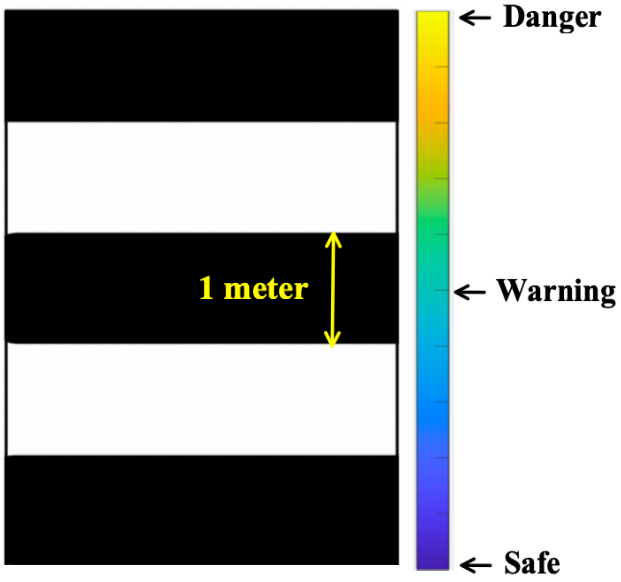


Figure 3.2: *Distance Meter with Heat Scale*

IV. Discussions

A. *Limitations*

Upon the building the system I had figured out that the concept I had thought of was a bit broader and needed a little bit more time to do research. Also, I noticed that if I had my own camera, I would've have been able to get more complex video content, things like lane curvature and changing of lanes. The main limitation is the limited availability of Matlab toolboxes. Toolboxes like computer vision would have helped the car detection stage.

B. *Improvements*

To officially make this system safe to use, a number benchmarking test is needed to get an overview of how accurate the distance meter and heat scale is when calculating the true distance. An attempt solve to the following issue would be to mark the road by meters and capture a set of images overlooking at the marked road. An image manipulation can be done to achieve an accurate record of distances.

V. Conclusion

After reaching the end of the project I have concluded that to accurately measure the distance from an object, the best approach is to use an actual Radar sensor. This paper proved that EBAS is possible using a camera but the accuracy of using such approach is very inaccurate.

REFERENCES

[1] Jason Brown, *Driving around suburban Norwich,* Feb. 18, 2016. Accessed on: December 10, 2019. [Video file]. Available: https://www.youtube.com/watch?v=n1bK1mNpYzg&t=57s

[2] MathWorks – houghlines*,, Extract line segments based on Hough transform.* Accessed on: December 12, 2019. [Online]. Available at: https://uk.mathworks.com/help/images/ref/houghlines.html

[3] *MathWorks* – imbinarize, *Binarize 2-D grayscale image or 3-D volume by thresholding.* Accessed on: December 12, 2019. [Online]. Available at: https://uk.mathworks.com/help/images/ref/imbinarize.html

[4] MathWorks – patch, *Plot one or more filled polygonal regions.* Accessed on: December 12, 2019. [Online]. Available at: https://uk.mathworks.com/help/matlab/ref/patch.html

[5] "Vehicle Detection and Distance Estimation - Milutin N. Nikolic – towards science," [Online]. Available: https://towardsdatascience.com/vehicle-detection-and-distance-estimation-7acde48256e1