On-Road Object Detection using Computer Vision Techniques for Advanced Driver Assistance Systems (ADAS)

Report Submitted for Review I

Sai Sravan Manne
Roll No. EDM13B018

Guided by:
Dr. Binsu J Kailath

Abstract

Advanced driver assistance systems (ADAS) are one of the fastest-growing segments in automotive electronics, some currently available ADAS are adaptive cruise control, automatic parking system, collision warning, autonomous navigation and so on. The current state of the art in ADAS is autopilot system i.e. a self-driving vehicle. Autopilot system in automobiles is expected to be the next disruptive technology in the upcoming decade. In autopilot system, real-time object classification is one of the major modules.

A typical set of objects that are seen on road are dogs, cats, cow, pedestrians, motorbikes, cars, trucks, trees, dustbins, garbage. In the past several algorithms were proposed based on edge, pattern, and texture recognition to identify these objects. But recently in 2012, a deep learning based algorithm called Convolutional Neural Network (CNN) [1][2] has been demonstrated to be a good classification technique achieving better accuracy in least computation time. However, the reduced computation time comes at the cost of increased Graphic Processing Unit(GPU) memory, thereby increasing the overall cost of the system. In this project, we discuss an algorithm that can add on to the existing CNN based solutions, and improve the computation time with minimal usage of GPU.

Objective

The chief objective is to develop an algorithm that can efficiently and accurately detect the on-road vehicles(both stationary and non-stationary). This entire project involves the following stages:

1. Understanding the various computer vision techniques currently being used for on road object detection.

2. Developing the initial design of the algorithm in MATLAB, and later implementation using Python with a major focus on optimizing its performance on GPU.

Work Done

A detailed study is made on the common characteristics of the images in KITTI Dataset[3], this information is used to simplify the previously developed algorithm. The following are a few notable things that appear in over eighty percent of the images available in the KITTI Dataset:

1. Most of the roads are one way.
2. Roads have proper solid white lane line markings.
3. Roads have proper broken white lane line markings.
4. A four road junction appears for not more than 8 frames.
5. Only fifty percent of the image frame depicts the road(due to the mounting position of the camera).

Basing on the above-mentioned points, the following techniques or methods are incorporated into the algorithm.

1. Since only fifty percent of the image frame depicts the road, the entire image frame is first truncated into two halves(horizontally).
2. The solid white lane line markings can be used to separate the road from its adjacent surroundings, this in turn greatly reduces the noise that has to be filtered in order to detect the on road objects.
3. The picture frame can be divided into two halves(vertically), and the dominant line in both the halves can be found out using Hough transform, this process is illustrated in Fig. 7a, 7b.
4. The dominant lines found out in each frame are initially stored and compared with the previous values in the database, a new line is accepted or considered as lane only if it appears in consecutive frames(8 frames). This technique is especially useful to nullify bad and missing lane markings in certain image frames.
5. During every five iterations, the lines with least number of repetitions are discarded.

6. A filter is developed that can reduce the effect of illumination variations and broken white lane line markings, the filter is discussed in detail in further sections.

Problems that were rectified due to the newly adopted techniques:

1. Reduction in the number of filters, thereby increasing the overall computation speed.
2. Reduction in the number of false detections, that were mainly caused due to signboards and other bright objects in image frames.
3. Discontinuity in object detections due to variations in surrounding brightness.
4. Failing to detect objects that have zero relative velocity(moving with the same speed and in the direction similar to the vehicle on which the camera is mounted).

Detailed description of the developed algorithm:

The proposed system is implemented on windows workstation with AMD GPU in MATLAB. The GPU has 2GB graphics memory and the workstation is powered by Intel i5 with 8GB RAM. The algorithm is initially developed in MATLAB because of the user-friendly interface and debugging techniques. The following is the brief description of the built-in functions being utilized:

1. **BlobAnalysis System Object:**

   The BlobAnalysis object computes statistics for connected regions in a binary image. It computes the following parameters for the connected regions: area, centroid, bounding box, major-axis, minor-axis, orientation, eccentricity, equivalent diameter, perimeter.

2. **Dilation:**

   The binary dilation of A by B denoted A⊕B, is defined as the set operation:

$$A \oplus B = \left\{ z \middle| (\hat{B})_z \cap A \neq \emptyset \right\}, \tag{1.1}$$

   where B is the reflection of the structuring element B. In other words, it is the set of pixel locations z, where the reflected structuring element overlaps with foreground pixels in A when translated to z. Note that some people use a definition of dilation in which the structuring element is not reflected.

In the general form of *grayscale dilation*, the structuring element has a height. The grayscale dilation of *A(x,y)* by *B(x,y)* is defined as:

$$(A \oplus B)(x,y) = \max\left\{ A(x-x',y-y') + B(x',y') \middle| (x',y') \in D_B \right\}, \qquad (1.2)$$

where D_B is the domain of the structuring element *B* and *A*(*x*,*y*) is assumed to be $-\infty$ outside the domain of the image. To create a structuring element with nonzero height values, use the syntax strel(nhood,height), where height gives the height values and nhood corresponds to the structuring element domain, D_B.

Most commonly, grayscale dilation is performed with a flat structuring element (*B*(*x*,*y*) = 0). Grayscale dilation using such a structuring element is equivalent to a local maximum operator:

$$(A \oplus B)(x,y) = \max\left\{ A(x-x',y-y') \middle| (x',y') \in D_B \right\}. \qquad (1.3)$$

3. **Erosion:**

The *binary erosion* of *A* by *B*, denoted $A \ominus B$, is defined as the set operation $A \ominus B = \{z|(B_z \subseteq A\}$.

In other words, it is the set of pixel locations *z*, where the structuring element translated to location *z* overlaps only with foreground pixels in *A*.In the general form of *gray-scale erosion*, the structuring element has a height. The gray-scale erosion of *A*(*x*, *y*) by *B*(*x*, *y*) is defined as:

$(A \ominus B)(x, y) = \min\ \{A(x + x', y + y') - B(x', y') \mid (x', y') \in D_B\}$, (1.4)

where D_B is the domain of the structuring element *B* and *A*(*x*,*y*) is assumed to be $+\infty$ outside the domain of the image. To create a structuring element with nonzero height values, use the syntax strel(nhood,height), where height gives the height values and nhood corresponds to the structuring element domain, D_B.

Most commonly, gray-scale erosion is performed with a flat structuring element (*B*(*x*,*y*) = 0). Gray-scale erosion using such a structuring element is equivalent to a local-minimum operator:$(A \ominus B)(x, y) = \min\ \{A(x + x', y + y') \mid (x', y') \in D_B\}$ (1.5)

4. **Hough Transform:**

The Hough Transform block implements the Standard Hough Transform (SHT). The SHT uses the parametric representation of a line:

$$rho = x * \cos(theta) + y * \sin(theta) \tag{1.6}$$

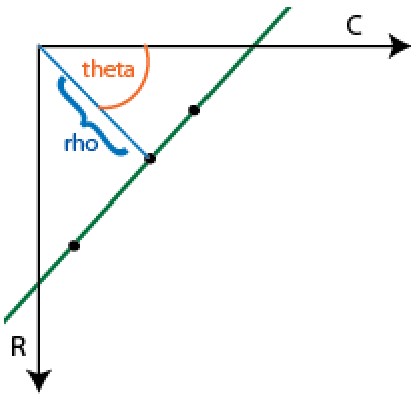

Figure 1: representation of rho and theta

The variable rho indicates the perpendicular distance from the origin to the line.
The variable theta indicates the angle of inclination of the normal line from the x-axis.
The range of theta is $-\frac{\pi}{2} \leq \theta < +\frac{\pi}{2}$ with a step-size determined by the Theta resolution (radians) parameter. The SHT measures the angle of the line clockwise with respect to the positive x-axis.
The Hough Transform block creates an accumulator matrix. The (rho, theta) pair represent the location of a cell in the accumulator matrix. Every valid (logical true) pixel of the input binary image represented by (R,C) produces a rho value for all theta values. The block quantizes the rho values to the nearest number in the rho vector. The rho vector depends on the size of the input image and the user-specified rho resolution. The block increments a counter (initially set to zero) in those accumulator array cells represented by (rho, theta) pairs found for each pixel. This process validates the point (R,C) to be on the line defined by (rho, theta). The block repeats this process for each logical true pixel in the image. The Hough block outputs the resulting accumulator matrix.

Level wise description and output of the Algorithm:

1. Input image frame to the algorithm is shown in Fig. 2.

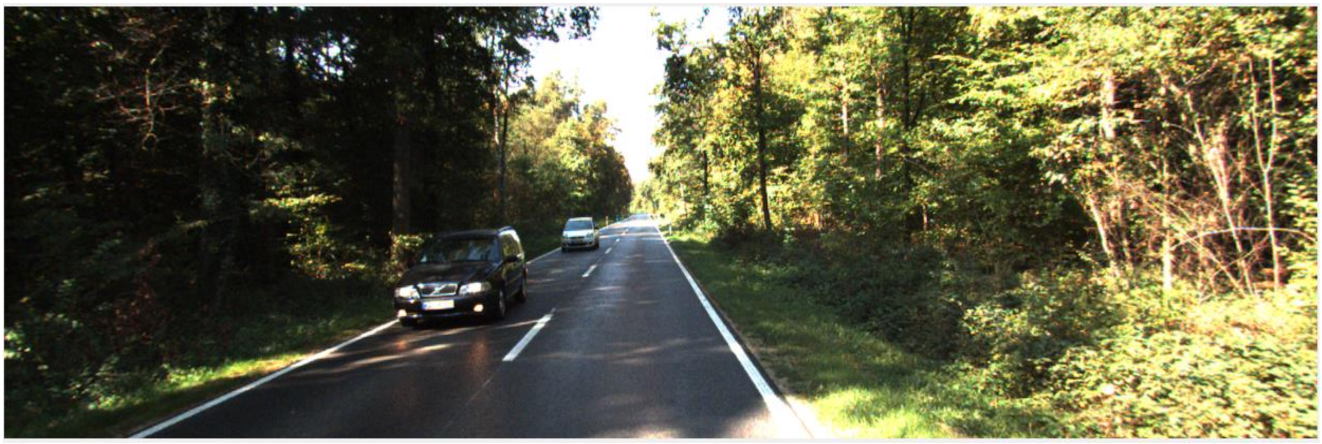

Figure 2: Input image frame to the algorithm

2. The truncated form of the input image is shown in Fig. 3.

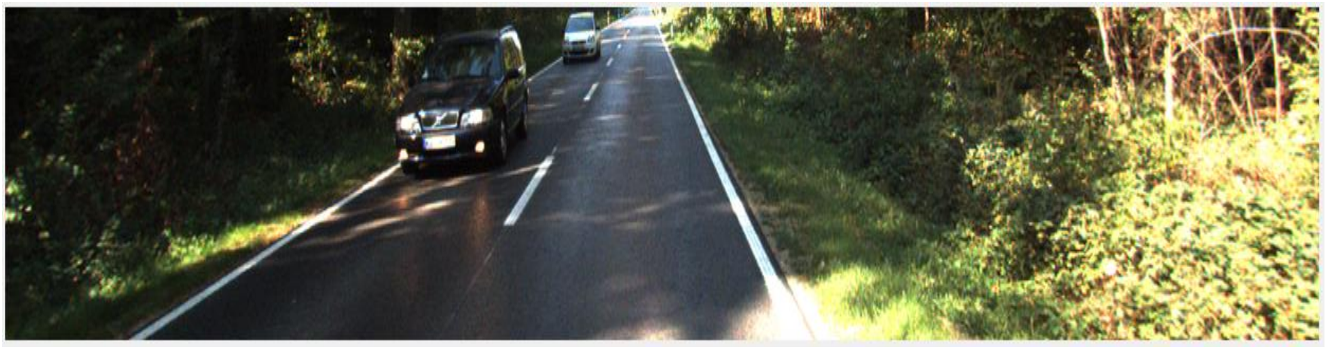

Figure 3: the truncated form of Fig. 2

3. The truncated image is converted into a two dimensional quantity as shown in Fig 4.

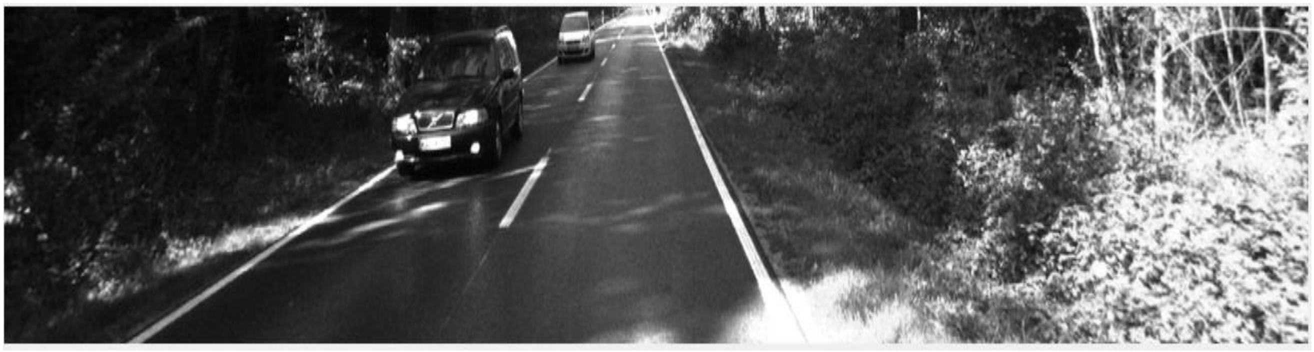

Figure 4: grayscale output

4. Edge detection filter is applied to find out the dominant edges in Fig. 4, the output of this operation is shown in Fig. 5.

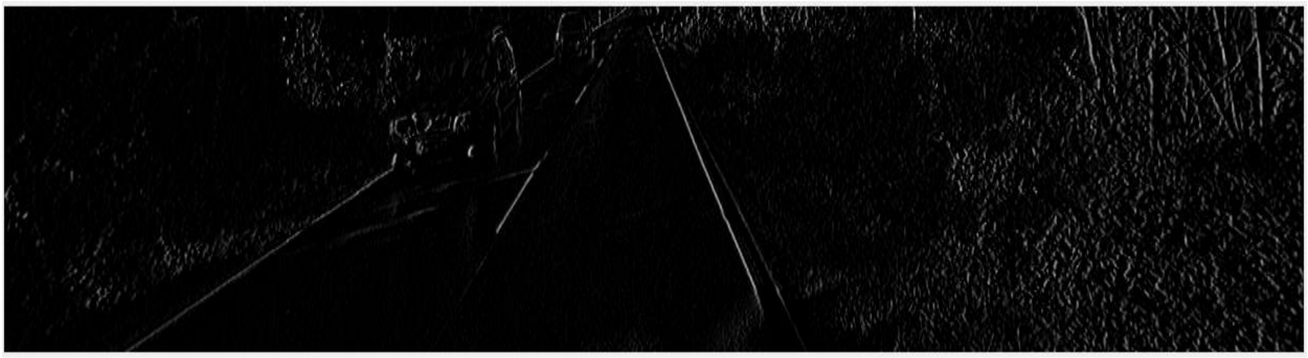

Figure 5: output from edge detection filter.

5. Now, Fig. 5 is subjected to a threshold operation where the values beyond the fixed threshold are treated as ones and the rest are made into zeros. The output of this operation is shown in Fig. 6.

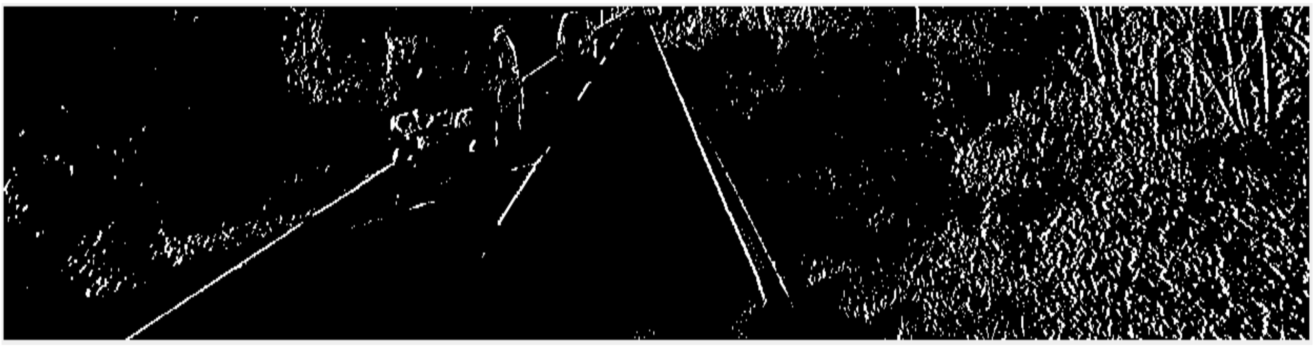

Figure 6: Output from threshold operation

6. Hough Transform is applied on the output in Fig. 6 and the output obtained is shown in Fig. 7a & 7b.

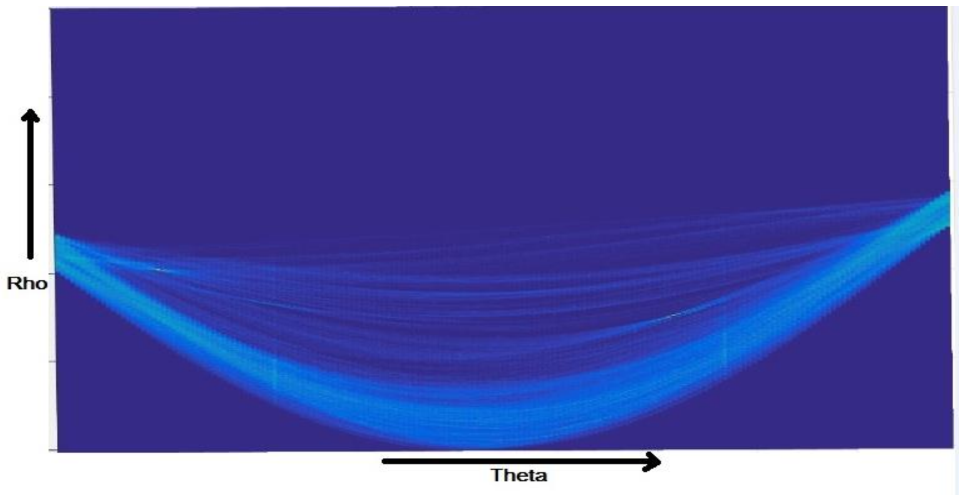

Figure 7a: plot of rho vs theta

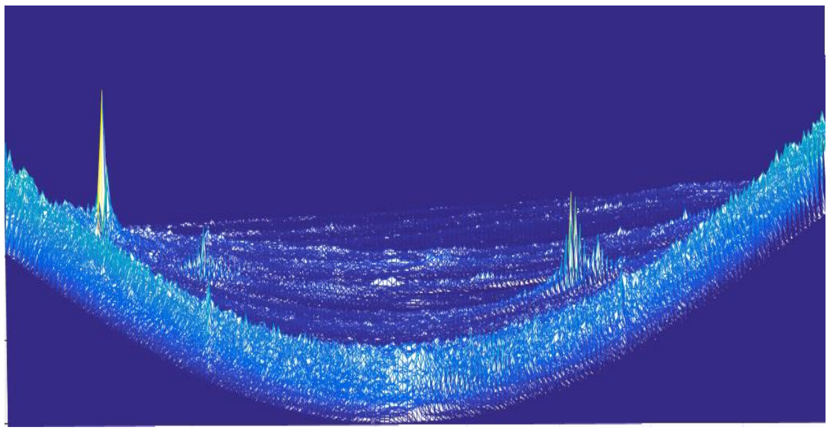
Figure 7b: frequency plot of rho vs theta

7. The rho and theta values having maximum frequency are found out from the peaks in Fig. 7b, these values correspond to the lane markings on the road. Then a polygon is drawn(pink colour) that fits in between the lines as shown in Fig. 8. The area occupied by the polygon corresponds to the area covered by the road.

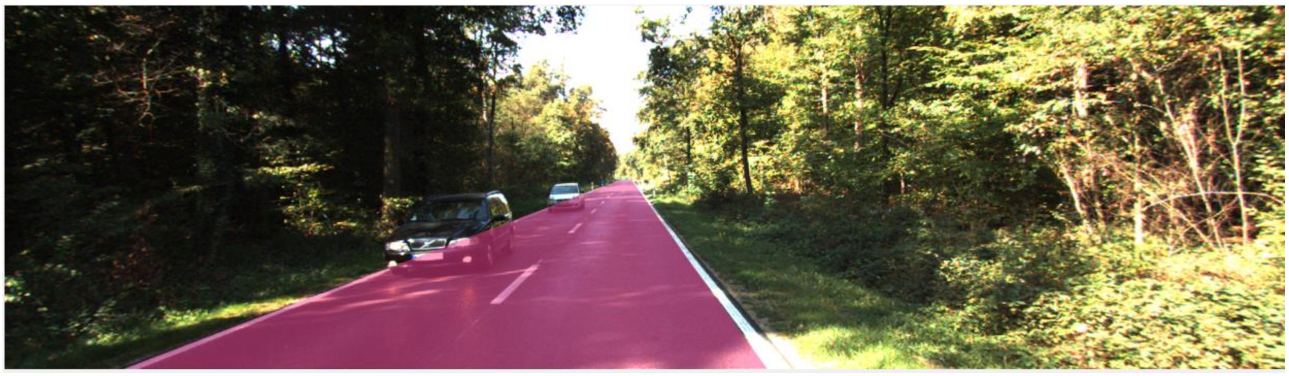
Figure 8: solid white lane line detection output

8. The road detected in Fig. 8, is separated from the rest of the image, the output of this operation is shown in Fig. 9.

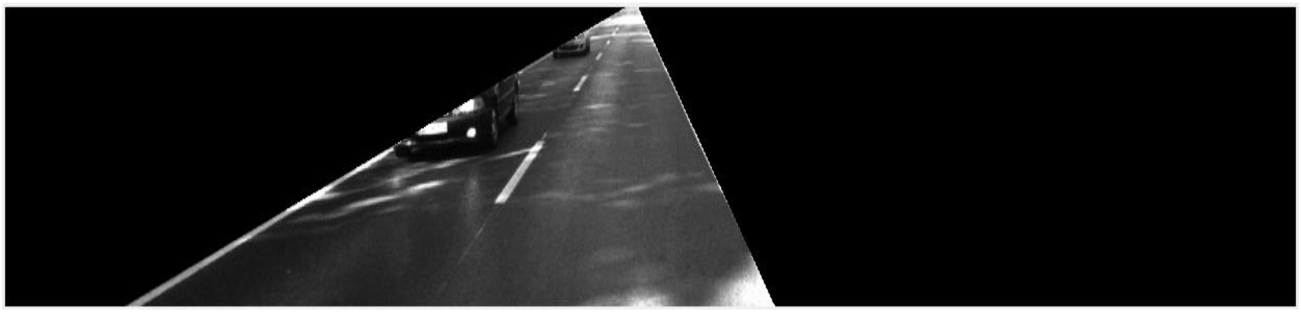
Figure 9: road separation from surrounding area

9. Fig. 10 displays the output of the built-in edge detection algorithm, it can be seen that the illumination variations on the road cause a lot of disturbances making it almost impossible to detect the vehicle.

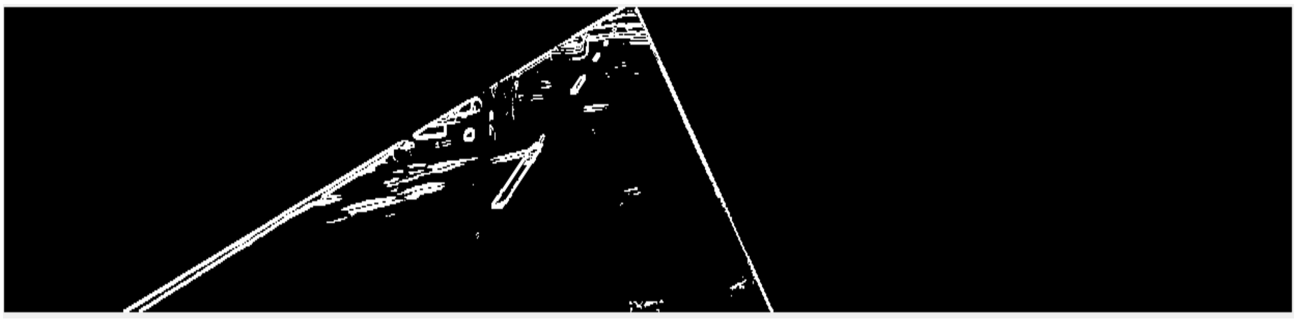
Figure 10: output from edge detection operation done on Fig. 9

10. In order to overcome the problem in step nine, a custom filter is developed to detect only the vehicles irrespective of the changes in illumination variations. The output of the filter can be seen in Fig. 11 .

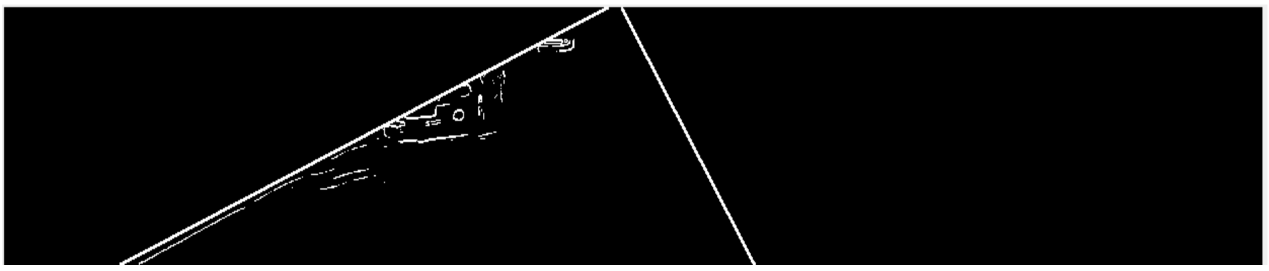
Figure 11: output from the custom designed filter

The designed filter smoothens the steep gradients in the colour patterns and at the same time compresses the values beyond a certain threshold(in this case it is 0.3). This process can be better understand by observing the following Fig. 12 a, b, c & d.

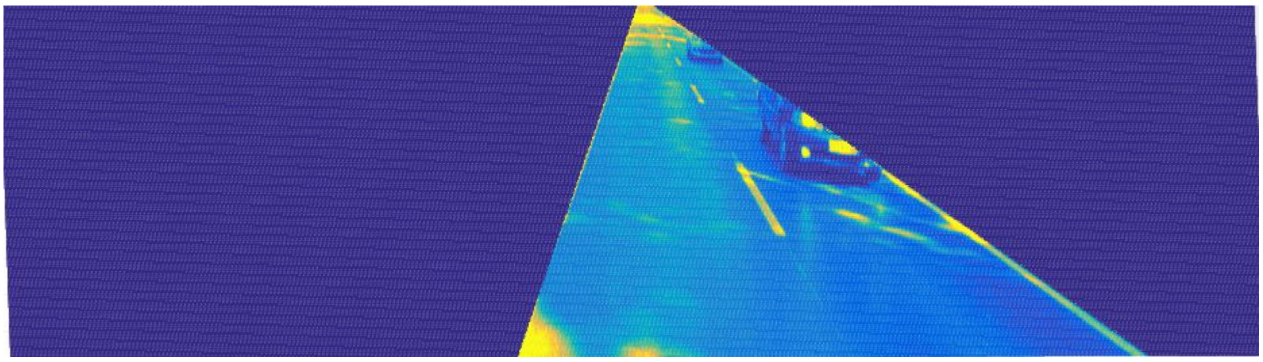
Figure 12a: brighter shades represented in yellow and darker shades are represented in blue.

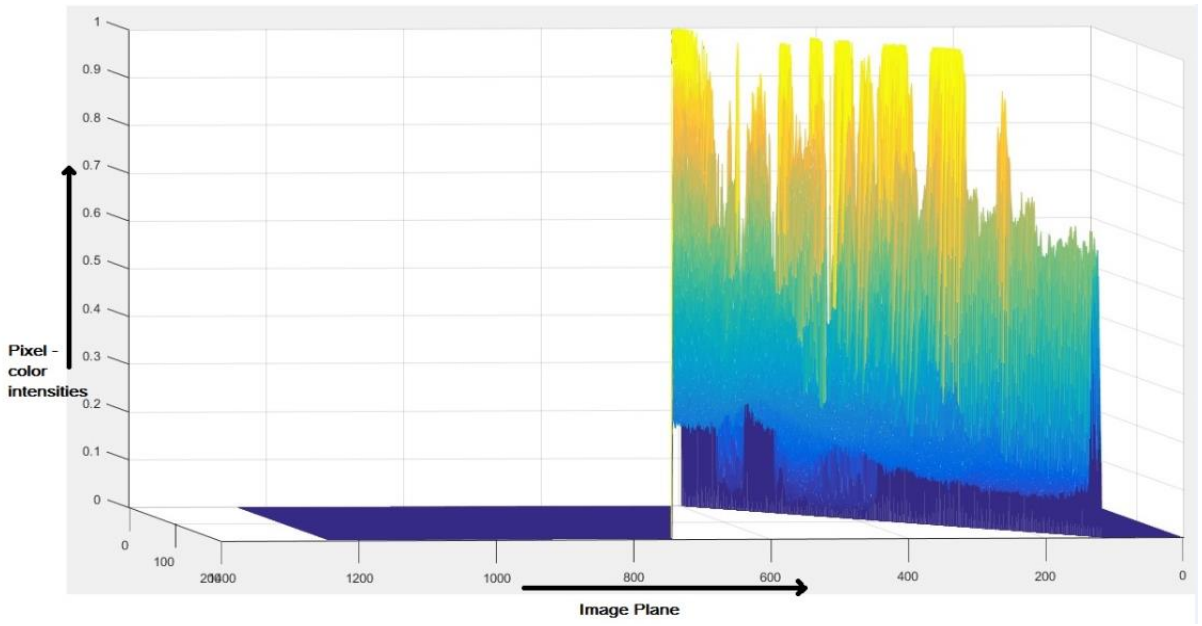

Figure 12b: variation in colour intensities

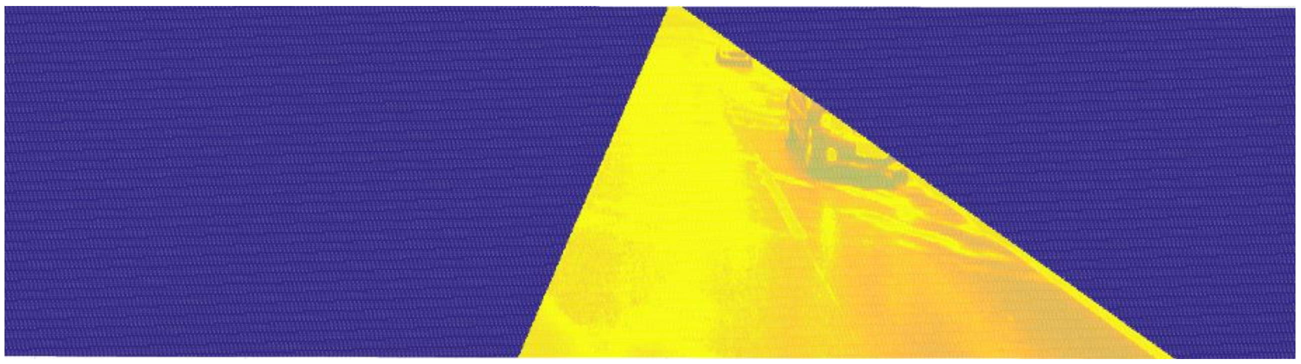
Figure 12c: output from the filter

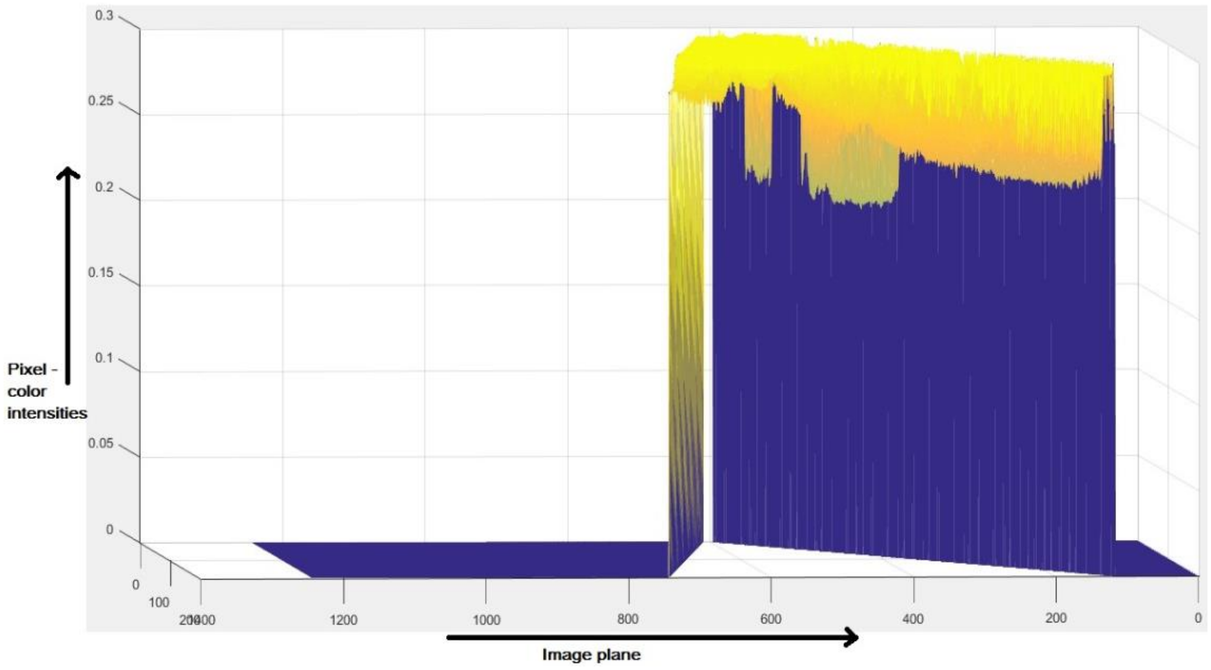

Figure 12d: reduced variation in colour intensities, max intensity is 0.3

11. The output shown in Fig. 11 is processed to remove the lane edges, this is being done in order to facilitate the erosion and dilation operations. The output of this operation can be seen in Fig. 13.

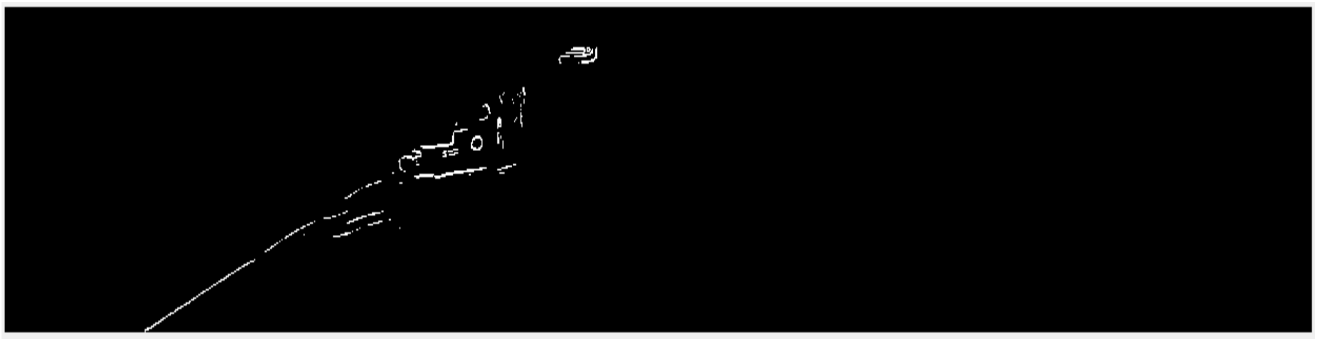
Figure 13: output after removing the lane edges

12. Area filtration(white spaces with an area less than 10 units is removed) and erosion operations are performed on the output in Fig. 13, this is being done in order to enlarge the area covered by the vehicles. The output of this operation can be seen in Fig. 14.

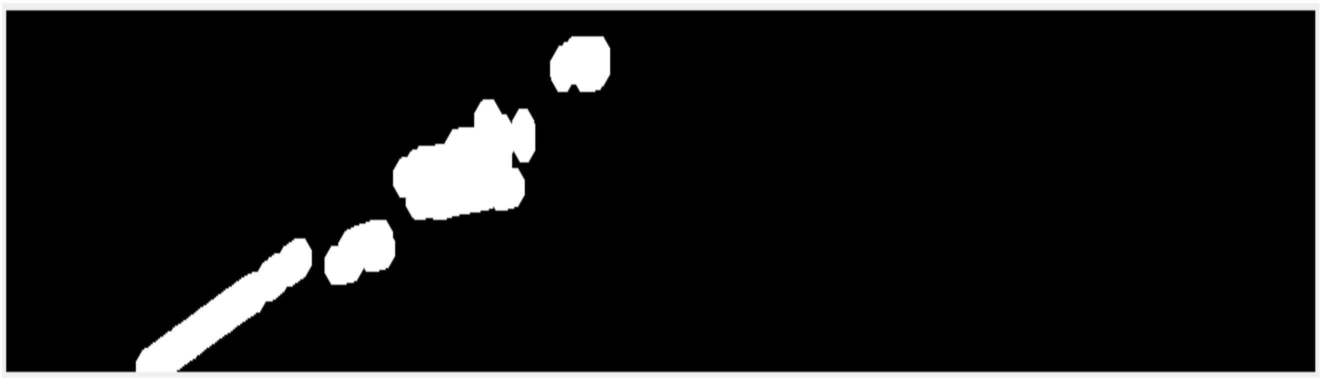
Figure 14: output from area filtration and erosion operations

13. Built in MATLAB function called Blob Analysis is being used to find out the area and centre of the white spaces in Fig. 14 . The output of this function is a matrix consisting of centre coordinates and the height and width of the bounding boxes.

14. The output of the function in step thirteen is further processed to track the detected white spaces(vehicles), the detected white spaces are further processed to remove false detections, however, the efficiency of the current method used isn't as expected. This has to be further improved to filter the unwanted detections. Bounding boxes are generated for the detected white spaces(vehicles), and the output is marked as shown in the Fig. 15.

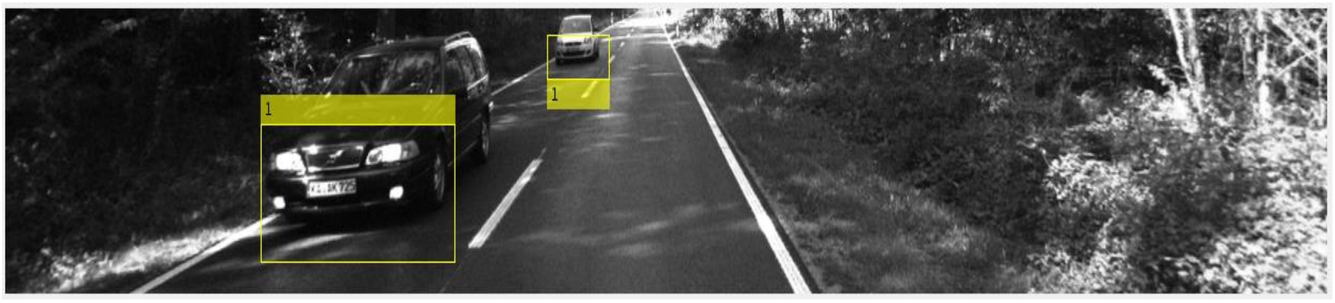

Figure 15: final output from the algorithm

Work to be done

1. The number of false detections has to be reduced, and the overall accuracy of the algorithm can be improved.
2. The entire approach has to be implemented in Python, with a major focus on optimizing its performance on GPU.

References

[1] Kai Kang, Hongsheng Li, Junjie Yan, Xingyu Zeng, Bin Yang, Tong Xiao, Cong Zhang, Zhe Wang, Ruohui Wang, Xiaogang Wang, and Wanli Ouyang, "T-CNN: Tubelets with Convolutional Neural Networks for Object Detection from Videos," published in Conference on Computer Vision and Pattern Recognition(CVPR) 2016.

[2] Andrej Karpathy, George Toderici, and Sanketh Shetty, "Large-scale Video Classification with Convolutional Neural Networks," published in Conference on Computer Vision and Pattern Recognition (CVPR) 2014.

[3] Andreas Geiger, Philip Lenz, Christoph Stiller and Raquel Urtasun, "Vision meets Robotics: The KITTI Dataset," published in International Journal of Robotics Research (IJRR) 2013.