ECE 515
COMPUTER VISION AND IMAGE ANALYSIS II
ADVANCED LANE DETECTION

AAKANSHA TUTEJA – 673835938
RUCHI RAJESH VAZIRANI – 652326908

ABSTRACT

Increasing safety and reducing road accidents, thereby saving lives are one of great interest in the context of Advanced Driver Assistance Systems. Among the complex and challenging tasks of future road vehicles is road lane detection or road boundaries detection. It is based on lane detection (which includes the localization of the road, the determination of the relative position between vehicle and road, and the analysis of the vehicle's heading direction). One of the principal approaches to detect road boundaries and lanes is using vision system on the vehicle. In this project, vision-based lane detection approaches capable of reaching real time operations are implemented. The system acquires the front view using a camera mounted on the vehicle then applies various vision- based techniques to detect the lanes. The first approach uses a linear regression which is fitting to the edges of the lanes detected using Hough transform to find the true lane edges. The second approach uses the distortion correction on images captured by the camera, then applies a curve fitting on the lane boundary detected images to extract the lanes. The proposed lane detection systems can be applied on both painted and unpainted road. The first approach works only on the straight roads or ones with slow curvature, while the second one works on straight as well as curved roads.

I. INTRODUCTION

Vehicle crashes remain the leading cause of accident deaths and injuries in Malaysia and Asian countries claiming tens of thousands of lives and injuring millions of people each year. Therefore, a system that provides a means of warning to the driver of the dangers has the potential to save considerable number of lives, the intelligent transportation system does the same by cooperating with the smart infrastructure to partially or fully automate the driving tasks for achieving a safer and better traffic environment. Among the automation tasks, the road detection plays a significant role in driving assistance systems that provides information such as lane structure and vehicle position relative to the lane. One of the main technology involves in these takes computer vision which become a powerful tool for sensing the environment and has been widely used in many applications by the intelligent transportation systems (ITS). In many proposed systems [1], the lane detection consists of the localization of specific primitives such as the road markings of the surface of painted roads. This restriction simplifies the process of detection, nevertheless, two situations can disturb the process: the presence of other vehicles on the same lane occluded partially the road markings ahead of the vehicle are the presence of shadows caused by trees, buildings etc. This project presents two vision-based schemes capable of reaching a real time performance in detection and tracking of structured road boundaries with slight and substantial curvature and is robust enough in presence of shadow conditions. In the first scheme the road boundaries are detected by fitting a linear regression model to the edges of the lanes after applying the Canny edge detection and Hough transform. The vehicle is supposed to move on a flat and straight or slow curvature road. For the second scheme, there are no such restrictions on the road conditions, as this scheme detects the road boundaries by fitting curve to the edges of the lane as detected after applying Perspective transform to the undistorted images as captured from the camera.

II. RELATED WORK

Currently many different vision-based road detection algorithms have been developed to avoid vehicle crash on the road. Among these algorithms Kreucher C. propose in [2] the LOIS algorithm as a deformable template approach. A parametric family of shapes describes the set of all possible ways that the lane edges could appear in the image. A function is defined whose value is proportional to how well a set of lane shape parameters matches the pixel data in a specified image. Lane detection is performed by finding the lane shape that maximizes the function for the current image. The Carnegie Mellon University proposes the RALPH system, used to control the lateral position of an autonomous vehicle [3]. It uses a matching technique that adaptively adjusts and aligns a template to the averaged scan line intensity profile to determine the lane's curvature and lateral offsets. The same university developed another system called AURORA which tracks the lane markers present on structured road using a color camera mounted on the side of a car pointed downwards toward the road [4]. A single scan line is applied in each image to detect the lane markers. LANA algorithm [5] was based on novel set of frequency domain features that captures relevant information concerning the magnitude and orientation of spatial edges extracted by 8*8(DCT). The GOLD system developed by Broggi, it uses an edge-based lane boundary detection algorithm [6]. The acquired image is remapped in a new image representing a bird's eye view of the road where the lane markings are nearly vertical bright lines on a darker background. Specific adaptive filtering is used to extract quasi vertical bright lines that concatenated into specific larger segments. A similar approach as to the GOLD system is also used in our implementations to form an extended equation to fit the road shape

III. ENVIRONMENTAL VARIABILITY

In addition to the intended application of the vision lane detection system, it is important to evaluate the type of conditions that are expected to be countered. Road markings can vary greatly not only between regions, but also over nearby stretches of road. Roads can be marked by well-defined solid lines, segmented lines, circular reflectors, physical barriers, or even nothing at all. The road surface can be comprised of light or dark pavements or combinations. An example of the variety of road conditions can be seen in Figure 1, some roads shows a scene with both solid line and dashed line lane markings. Lane position in this scene can be considered relatively easy because of the clearly defined markings and uniform road texture. But in another complex scene in which the road surface varies and markings consist of circular reflectors as well as solid lines the lane detection will not be an easy task. Furthermore, shadowing obscuring road markings makes the edge detection phase more complex. Along with the several types of markings and shadowing, weather conditions, and time of day can have a significant impact on the visibility of the road surface as shown in Figure 1 while Figure 2 shows some traffic sign recognition scenarios which act as hurdle in achieving highly efficient recognition system.

Figure 1 Example of extreme situations to be handled by lane detection system, (a) different lane markings; (b) Lane Occlusions; (c) different road textures; (d) Lanes with shadow; (e) Rainy road; (f) Saturated image at tunnel exit.

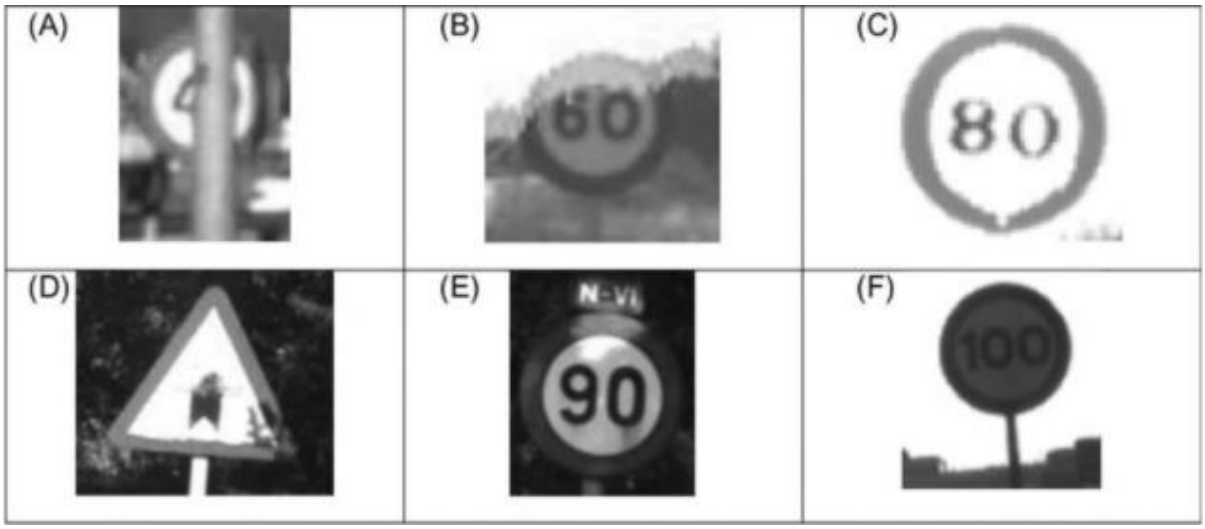
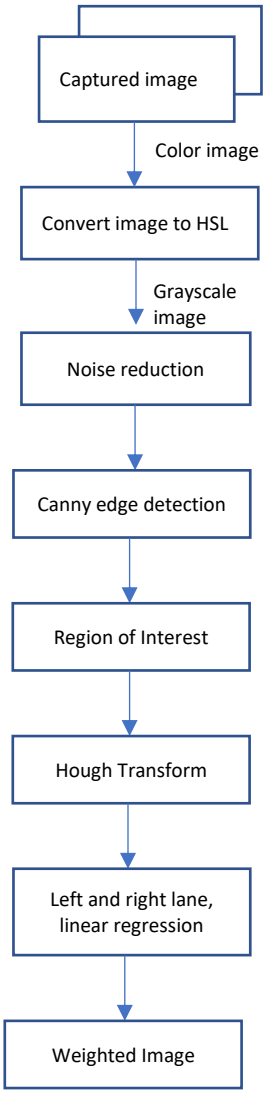
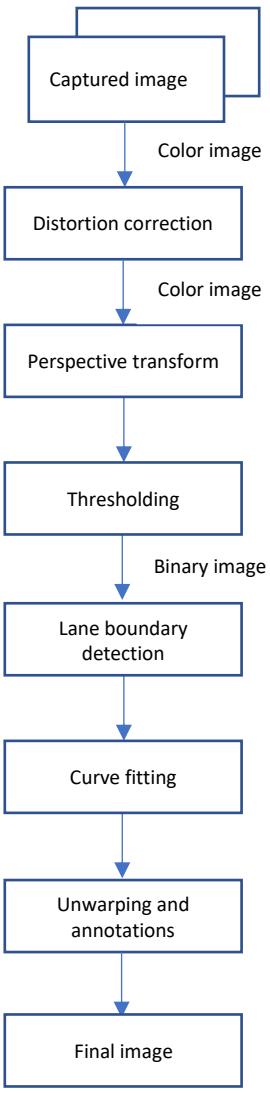


Figure 2 Examples of situations difficult to handle by traffic sign recognition system, (a) Occlusion; (b) Smearing; (c) Saturation; (d) Degradation; (e) Shadows; (f) Backlight

IV. OVERVIEW OF OUR IMPLEMENTATIONS



Implementation 1



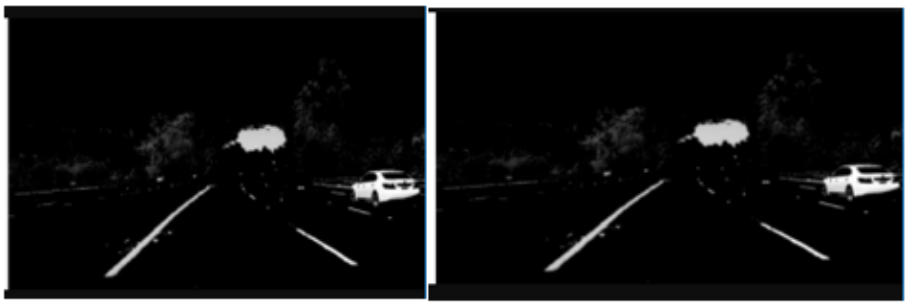
Implementation 2

- **IMPLEMENTATION 1**

Grayscale image and noise reduction

The input is color image sequence as captured by a color camera mounted inside the vehicle along the central line. The images are then converted to the HSL color system (H – Hue, S – Saturation and L - Lightness) and then to grayscale images. The need for conversion to HSL system is that if the image is converted to grayscale directly from RGB color system, a similar response is noticed for yellow and white lines while our aim is to differentiate between yellow and white road markings too as they both represent two different things. To filter out white color the S channel was thresholded and for yellow color the H channel was thresholded at value 30. Once the image is converted to a grayscale image, the next step was noise reduction which was done using a 3x3 Gaussian kernel. Figure 3 below shows the steps involved in this step

(a) (b)



(c) (d)

Figure 3 showing the steps involved in creating the Grayscale image, (a) original RGB image; (b) HSL converted image; (c) Thresholding; (d) Noise reduction using Gaussian kernel

Canny Edge Detection

The Canny transform is a two steps algorithm that looks for the intensity gradients using 4 filters i.e. horizontal, vertical and diagonal and high and low threshold as the parameters for categorizing the strength of the gradient. If the gradient is more than the high threshold it is categorized as a strong pixel gradient, if less than the low threshold the pixels are suppressed and if the value of the gradient is between low and high threshold it is categorized as a weak gradient. In next step the weak gradients are analyzed, if they come from a true edge, they are connected to a strong gradient, the ones which remain unconnected are dropped and could have arose from color variations or noise. In our implementation same thing happens and we try to find all the edges including the ones coming from the trees and other environmental factors. Figure 4 shows the output of the Canny edge detection in our implementation

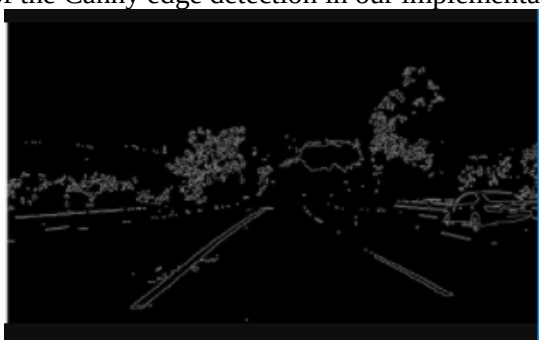
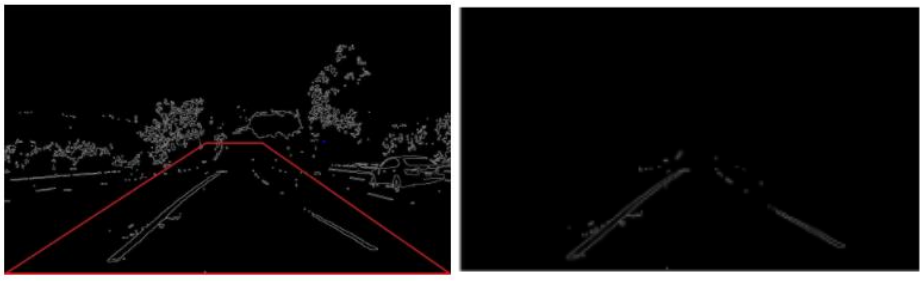


Figure 4 showing the result of Canny edge detection

Region of Interest

Once the edges have been detected by the Canny transform, an image mask is used on the resultant image to get the region of interest in the image. The ROI chosen here is trapezoidal in shape and helps us in getting rid of the unwanted environmental information like trees and other surroundings thus, reducing the calculations. The parameters used to form the ROI in the image mask are height, top and bottom width of the trapezoid as looks as shown in Figure 5



(a) (b)

Figure 5 showing the ROI extraction, (a) trapezoidal image mask that was used; (b) resultant of ROI image masking

Hough Transform

The Hough transform is used to find all the straight lines in any given image. Here it is applied to the ROI formed image using the OpenCV library function. When using this function in real applications, the Hough space of an image is divided into uniform clusters. The grid is defined by parameters which can be called by rho resolution and theta resolution. They often equal 1 pixel and 1 degree respectively. A sweep is then made through all the cells in the grid to count votes for a certain line with a specific slope. If there are more votes than a given threshold the straight line is claimed to be found and is described by rho and theta parameters. Other than rho, theta and threshold two other parameters named min line length and max gap between lines are used to make sure the lines corresponding to the lanes as shown in Figure 6 are detected

Figure 6 showing the output of Hough transform, detecting the straight lines

Linear Regression and Weighted Image

The Hough lines once detected are then filtered out based on slopes and end-point locations. This step is needed because the Hough transform detects the vehicle boundary as one of the possible edges of the lane which is not so, to get rid of them and similarly originated edges which are not a part of the true lane edges, we need the filtering. Only the edges with the slope lying between 17 – 56 degrees are retained and rest all others are suppressed. The edges are then classified into the left and right lane. Then a linear regression model is used on the edges detected to get the line equations of the lanes, which are then drawn on the binary image. All these operations are performed on a binary or grayscale image and the resultant image is also binary, we need to go back to the original colored image and create a weighted image by adding the original image before any processing with the resultant image which has the lane lines drawn on it. The output image looks like as shown in figure 7

Figure 7 showing the output image of Implementation 1

We faced a few problems in this implementation like overfitting and poor detection of curves. Hence, we decided to try out a new method.

- **IMPLEMENTATION 2**

This implementation uses a curve fitting technique instead of linear regression to detect the lanes with substantial curvature. The input for this one is also color images as captured from a camera mounted in front of the vehicle along the central line.

Camera Calibration and Distortion Correction

The input for this system is also RGB colored images in the raw form as captured by the camera, and since the images are captured by a wide-angle lens it is possible that the images suffer distortion. A distortion correction technique is to be applied on all the captured images to obtain quantitative information and accurate relationship between the 3-D object and its counterpart in a 2-D image. It is a two-step process, first one is to calibrate the camera. Here, we calibrated the camera against chessboard images captured at different angles as shown in Figure 8. First those images are converted to a grayscale image then OpenCV function findChessboardCorners() is used for finding the corners on a 9x6 chessboard, and then calibrateCamera() function is used to calculate the coefficients of the distortion matrix. Once the coefficients are calculated, all the images captured by the camera are undistorted using undistort() function as shown in Figure 9 so that we get exact location of the 3-D object in the 2-D image too.

(a) (b) (c)

Figure 8 showing the calibration against 9x6 chessboard, (a) distorted original image; (b) finding the corners; (c) undistorted image.

(a) (b)

Figure 9 showing the undistortion on original image, (a)original image; (b) undistorted image

Perspective transform

Once the images have been undistorted using the distortion matrix coefficients, perspective transform is applied on all the images/ each frame, to get a bird's eye view of the road which focuses only on the lane lines and displays them in such a way that they appear to be parallel to each other. This step saves a lot of memory space as the calculations have been reduced as only the lane boundaries remain as shown in Figure 10 and making it easier later to fit the polynomials to the lane lines and measure the curvature of the road.

Figure 10 showing bird's eye view of the lane boundaries.

Thresholded Image

After the perspective transform, we get an image which only has the lane boundaries in it, now we convert that resultant image into a binary image. For doing so, we convert the image into distinct color systems, threshold them on different channels and then combine them all to create a final threshold image. Threshold the S channel after converting the image in the HSL color system in the range 180- 255(giving us information about both yellow and white color), B channel in the LAB color system ranging 155- 200(to get the maximum information about the yellow road markings) and L channel in the LUV system ranging 225 – 255 to get the information about the white color and finally add these thresholded images to get an resultant binary image with the total information about white and yellow color road markings as shown in Figure 11.

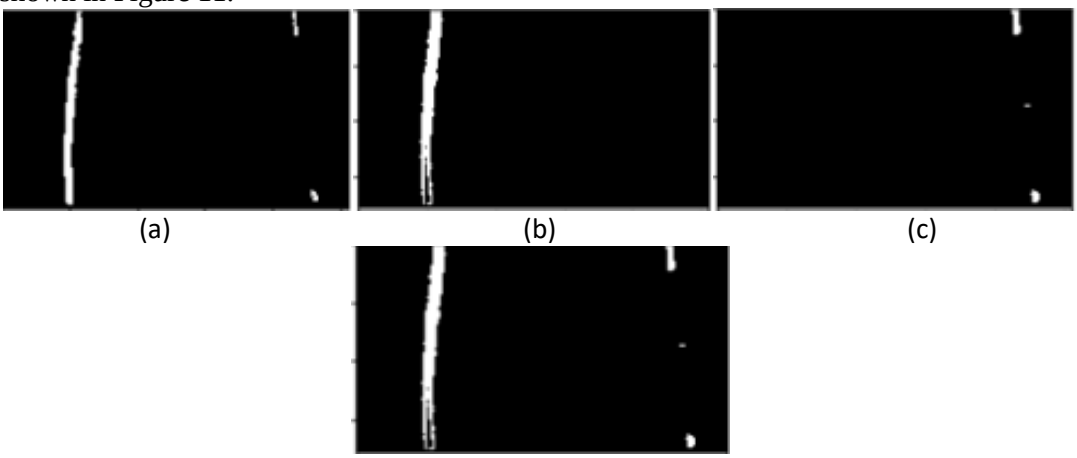


Figure 11 showing the formation of thresholded image, (a) S channel thresholded; (b) B channel thresholded; (c) L channel thresholded; (d) combined color thresholded image

Lane Boundary detection and Curve fitting

The thresholded image created from the above step is then processed to find a second order polynomial that fits the curve of the lane boundaries. For this, first a histogram of the bottom part of the image is made because in each image the bottom half gives the information about the lane boundaries. The histogram has two peaks corresponding to left and the right lane boundaries, which is then split vertically into two and horizontally into 9 slices. A window of 200 pixels is made which goes up each slice looking for the lane boundary pixels. Once the lane boundary pixels are found, a second order polynomial curve fitting is applied to it as shown in Figure 12, which further helps in calculating the radius of curvature of the lane. This process works fast when input is just the RGB images, but since here we tried to implement this solution on the real time-based lane detection problem for which the input is video it turned out to be slow. So, to make the implementation a bit robust the temporal correlation between the frames of a video has been exploited to achieve better response to the video input.

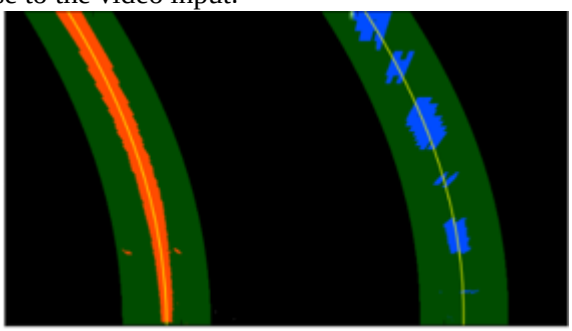


Figure 12 showing the 2nd order polynomial fit on the lane boundaries

Given the polynomial fit calculated from the previous video frame, search +/- 100 pixels horizontally from the previously predicted lane lines. Then we simply perform a 2nd order polynomial fit to those pixels found from our quick search. In case we don't find enough pixels, we can return an error (e.g. return None), and the function's caller would ignore the current frame (i.e. keep the lane lines the same) and be sure to perform a full search on the next frame. Also, to make the detector more robust to the noisy input, a moving average of the polynomial coefficients (3 values per lane line) for the most recent 5 video frames, this helps in smoothing out the effect of the noise.

Radius of curvature and vehicle offset

After having calculated the polynomial fit to both right and left lane boundaries, the radius of curvature is calculated using the formula mentioned below,

$$\text{Radius of curvature} = \frac{\left[1 + \left(\frac{dy}{dx}\right)^2\right]^{3/2}}{\left|\frac{d^2y}{dx^2}\right|}$$

A scale was assumed to convert the distance units from the pixels to meters, 30m per 720 pixels in the vertical direction and 3.7m per 700 pixels in the horizontal direction. Finally, the radius of curvature was averaged for left and right lanes and reported in the final image. For calculating the vehicle offset from the center, the center of the vehicle is assumed to be the point where the camera has been mounted. To find the center of the lane, an average of x bottom pixels of the left and right lane and then the vehicle offset is give by the difference between the vehicle center and the average of the bottom pixels.

Unwarping and final image

All the steps as mentioned above have been performed on a thresholded binary image, we need to go back to the original colored image. We start with a blank image and draw the polyfit lines as estimated by the detector and fill the region between those lines as the lane. Now we used inverse warping to go back to the original image from the bird's eye view, which was done using the inverse warping matrix calculated from the perspective transform and then all the annotations along with the text are overlaid over the original image as shown below in figure 13

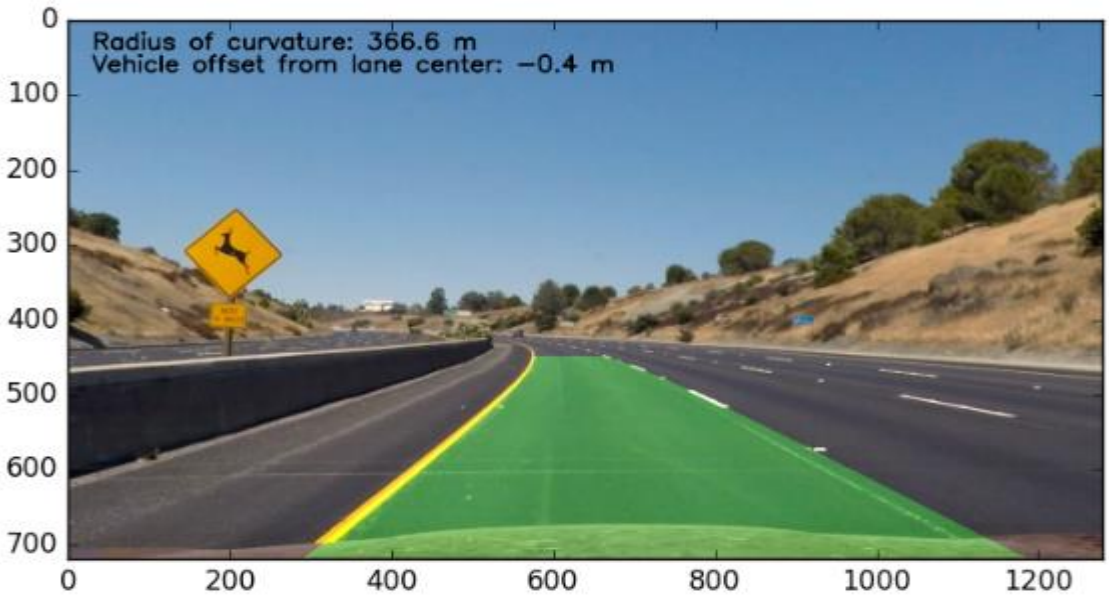


Figure 13 showing the output of Implementation 2

V. Conclusion

We tried to device schemes capable of reaching a real time performance in detection and tracking of structured road boundaries with slight and substantial curvature and is robust enough in presence of shadow conditions and inferred the following:

- Implementation 1 is overfitted to the available data and fails to detect lanes in different lighting conditions. Also, when the roads have a significant radius of curvature this system fails as it is trying to extrapolate only the straight lines. Thus, we tried to overcome some of them by our second implementation.
- Implementation 2 solves the problem of overfitting but detection gets affected when the light conditions change. It doesn't fail during sharp turns. Occlusions are still a problem in this method, Also, it is less robust.
- In the future, we'll try to device a scheme which doesn't get affected by lighting conditions and is more robust.

VI. References

[1] http://www.themalaysian.blogspot.com/2006/08/fatal-roadaccidents-ranking-malaysia.html, August.2006.
[2] C. Kreucher, S. K. Lakshmanan, “A Driver warning System based on the LOIS Lane detection Algorithm”.
[3] Pomerleau D. and Jochem,”Rapidly Adapting Machine Vision for Automated Vehicle Steering”, IEEE, 1996.
[4] M. Chen., T. Jochem D. T. Pomerleau, “AURORA: A Vision-Based Roadway Departure Warning System”.
[5] C. Kreucher and S. Lakshmanan, “LANA: a lane Extraction algorithm that uses frequency domain features”.
[6] B. M, Broggi, “GOLD: A parallel real-time stereo Vision system for generic obstacle and lane detection”.
[7] Abdulhakam.AM.Assidiq, Othman O. Khalifa, Md. Rafiqul Islam, Sheroz Khan “Real Time Lane Detection for Autonomous Vehicles”
[8] Yue Wang, Eam Khwang Teoh, Dinggang Shen “Lane detection and tracking using B-Snake”