LANE DETECTION USING DIGITAL IMAGE PROCESSING

17012041027-PRIT SHAH
17012041028-JAYGOPAL SHARMA
17012041029-SHREYANSH JAIN

2MC706 MINOR PROJECT

Guided by: - Shri TRUSHAR SHAH
Department of Mechatronics Engineering
U.V. PATEL COLLEGE OF ENGINEERING
GANPAT UNIVERSITY

CERTIFICATE

Towards partial fulfillment of requirement for the award of Degree of "Bachelor of "**MECHATRONICS ENGINEERING"** of "**GANPAT UNIVERSITY"**. This is the record of candidates own work carried out by them under our supervision & guidance. In our opinion the work submitted has reached a level required for being accepted for exam. The matter embodied in this project has not been submitted to any other university or institute.

Project Report on "LANE DETECTION USING DIGITAL IMAGE PROCESSING"

Guided by:
Shri TRUSHAR SHAH
Assistant professor
Mechatronic Department
U.V. Patel College of Engineering

Submitted By:
Prit Shah(17012041027)
Jaygopal Sharma(17012041028)
Shreyansh Jain(17012041029)

Contents

ABSTRACT

In developed countries, the automation and intelligent development of vehicles has reached a relatively high level and has gradually developed in many countries. In recent years, algorithms for lane detection have emerged in an endless stream, but the advantages and disadvantages of comprehensive comparison of various algorithms have resulted in the following problems: First, the robustness of lane line detection is poor, mainly due to the surrounding environment of the road. The impact is greater, in traffic-intensive city streets, affected by natural factors such as trees or building shadows around the driveway; second, the real-time nature of the lane line detection is poor, affected by other marking lines on the driveway, or damaged in the lane. When the pollution is serious, the image of the lane line collected by the system is incomplete and of poor quality, which increases the difficulty of analysing and processing data of the detection system.

This article aims at the problems existing in the current lane line detection and determines the research focus of the article:

(1) Improve the accuracy and real-time performance of the detection algorithm. Most of the factors affecting the detection of the lane line are generated before the image is acquired. This requires the strict pre-processing of the collected lane line image to remove a large amount of interference. After the information, not only can the detection result be more accurate but also the complexity of the algorithm be simplified, making the detection result accurate and effective.

(2) This paper simulates the detection algorithm from two aspects of PYTHON, OpenCV, and CARLA .This method can effectively improve the image processing speed, save the logic resources, and better realize the lane recognition function.

(3) Improve the existing lane line detection algorithm. There are many algorithms for lane detection, but each algorithm has its own advantages and disadvantages. Part of this article summarizes the advantages and disadvantages of these detection algorithms, analyzes their feasibility in actual detection, and improves the algorithm based on this.

ACKNOWLEDGEMENT

We would like to thank our project guide **Shri Trushar Shah** for his support and conceptual help at various technical problems.. The very concept of the project as well as its realization would not have been possible without him.

Lastly we would like to thank all the faculty members of **Mechatronics department** for their support throughout the year.

Apart from our faculty members we've also gained a lot of experience and expertise from interacting with our classmates and friends. We would like to thank them all for being with us in this journey and making it memorable.

1. INTRODUCTION

In today's era the concept of self-driving car is increasing at an enormous rate. With the constant popularity of automobiles and the continuous improvement of people's living standards, almost every family owns one or more cars, providing people with a comfortable and convenient travel life. Although entering the twenty-first century, the post-modern transportation system has been very developed, but it is still not in direct proportion to the increasing traffic environment demand of people. The rapid development of the automotive industry has also brought with it environmental pollution, energy shortages, traffic jams, and traffic safety issues, among which traffic safety issues are particularly acute and thorny and have developed into a major global problem[3]. To make self-driving car road friendly, lane detection plays an essential role in this. In this project we have built our own lane detection using Digital Image Processing (DIP) with the help of Python and OpenCV.

2. PROBLEM FORMULATION

As increase in driving error, lack of attention and other common mistakes by human being there is increase in the car or vehicle accidents. To solve this problem automation is needed in driving vehicle because it is considered to be the mundane task.

3. OBJECTIVE

The purpose of our project is to detect lane on pre-recorded tract and also on simulator. In our project we have taken the reference of CARLA as simulator.

4. CARLA

In this paper, we introduce CARLA (Car Learning to Act) – an open simulator for urban driving. CARLA has been developed from the ground up to support training, prototyping, and validation of autonomous driving models, including both perception and control. CARLA is an open platform. Uniquely, the content of urban environments provided with CARLA is also free. The content was created from scratch by a dedicated team of digital artists employed for this purpose. It includes urban layouts, a multitude of vehicle models, buildings, pedestrians, street signs, etc. The simulation platform supports flexible setup of sensor suites and provides signals that can be used to train driving strategies, such as GPS coordinates, speed, acceleration, and detailed data on collisions and other infractions. A wide range of environmental conditions can be specified, including weather and time of day.

Environment:- The environment is composed of 3D models of static objects such as buildings, vegetation, traffic signs, and infrastructure, as well as dynamic objects such as vehicles and pedestrians. All models are carefully designed to reconcile visual quality and rendering speed: we use low-weight geometric models and textures, but maintain visual realism by carefully crafting the materials and making use of variable level of detail. All 3D models share a common scale, and their sizes reflect those of real objects.[5]

Figure 1: A street in Town 2, shown from a third-person view in four weather conditions. Clockwise from top left: clear day, daytime rain, daytime shortly after rain, and clear sunset. See the supplementary video for recordings from the simulator.

5. METHODOLOGY

a. Cropped image

Cropping is a procedure that is used to eliminate the unwanted region from a particular image. During the determination of lane lines, we only need to focus on the regions where we are likely to see the lanes. For this reason, the cropping operation is done and performing the further image processing only in the particular areas of the image. The cropped image to focus the particular region of lanes is illustrated in Fig. 3. We have resized the image to smaller dimensions. This supports with doing the image processing pipeline faster.

b. Edge detection

Commonly used edge detection algorithms include global extraction methods based on energy minimisation criterion (such as fuzzy theory and neural networks) and edge derivative methods that use differential operators (such as Canny, Prewitt, LOG and Robert operators). However, the traditional single-edge detection algorithm has many problems, including a too-wide detection range, poor antinoise interference and prolonged calculation time . Meeting the target requirements of lane detection is difficult. To overcome the shortcomings of the algorithms listed above, this study adopted the edge detection algorithm, which superimposes the threshold of the Sobel operator and HSL colour space.

The Sobel operator is a type of discrete first-order difference operator. The influence of adjacent points in the neighbourhood is different for the current pixel point. Greyscale weighting and differential operations were performed on neighbourhood pixels to obtain the gradient and normal vectors of the pixel point that can be used to calculate the approximate value of the image brightness function.

The Sobel operator performs edge detection on the detected image from the horizontal and vertical directions and can sufficiently filter the interference noise with the improved image processing effect. However, if only the Sobel operator is used for edge detection, then problems such as low detection accuracy and rough edges easily occur. These issues must be solved effectively. Figure(a) shows the HSL colour space, which can be converted from the common RGB colour space. In this figure, H, S and L denote the hue, saturation and lightness, respectively. H represents the colour change of the image. The position of the spectral colour is represented by the angle, and different colour values correspond to different angles. S refers to the colour degree of the image colour used to describe the similarity between the actual and standard colours. When S is the minimum value of 0, the image becomes a greyscale image and H is undefined. When S is the maximum value of 1, the image colour and its complementary colour are fully saturated. L indicates the lightness degree of the image colour, which gradually changes along the direction of the axis. The maximum and minimum values are 1 and 0 in white and black, respectively.

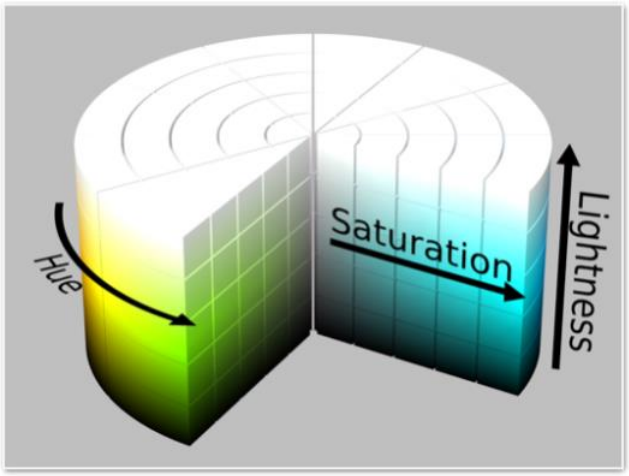
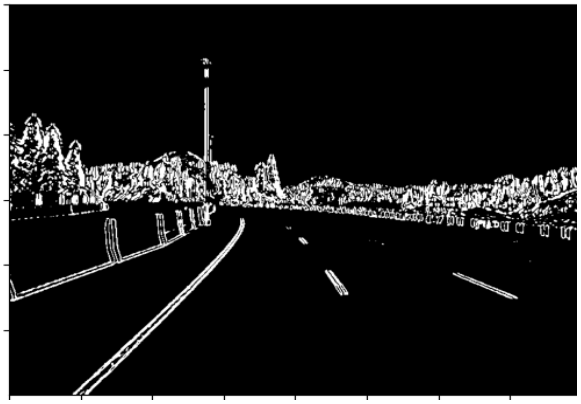


Figure (a) :The HLS color space

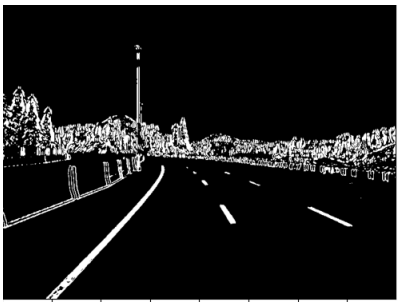
Compared with the RGB colour space, the HSL colour space can better reflect the visual field perception characteristics of the naked eye. Each colour channel in the HSL colour space can be processed independently and separately. Among them, the component image with S channel performs best and is beneficial to reducing the workload of image processing remarkably and significantly improving the target recognition rate of the detected image.

Figure(b)



Figure(c)

i. Perspective transformation



Figure(d)

Figure(e)

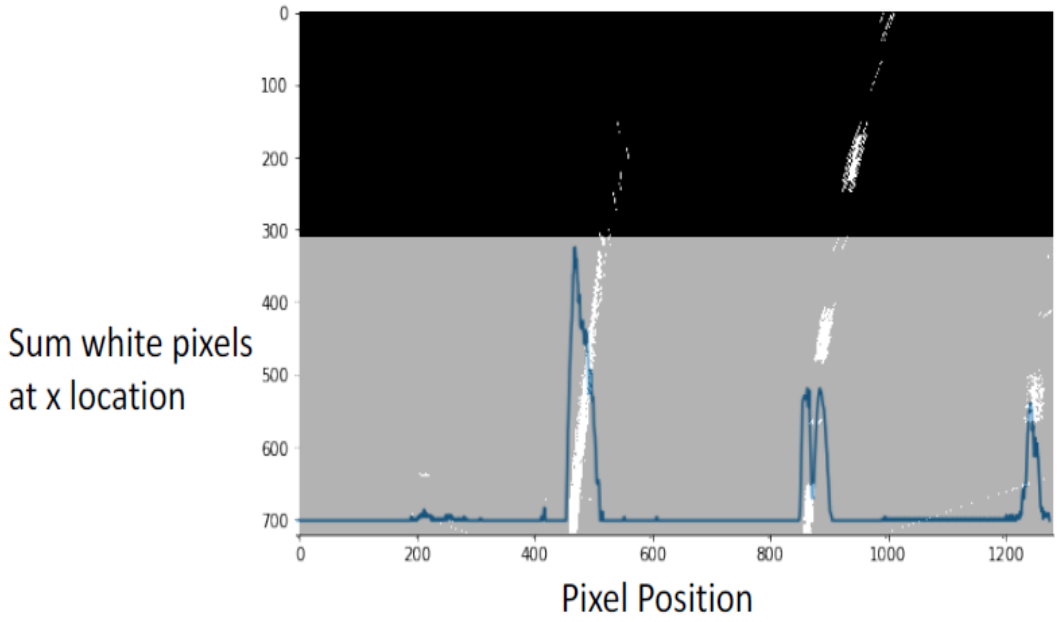
When human eyes see near things they look bigger as compare to those who are far away. This is called perspective in a general way. Whereas transformation is the transfer of an object etc. from one state to another.

So overall, the perspective transformation deals with the conversion of 3d world into 2d image. The same principle on which human vision works and the same principle on which the camera works. As you can see that there are two images (d) and (e). Image (d) is an image from camera view and image (e) is perspective view[1]. To do this we need 4 sets of points with coordinates in the original image. This method allows to use images of any size, and relative points will not change.

This means that source points will fill the whole area of destination image. Destination image will have the same dimensions as the source image. The method uses *cv2.getPerspectiveTransform()* to get transform matrix[2].

c. Advanced techniques for lane finding

I. Histogram



Figure(f)

A histogram is a graph. A graph that shows frequency of anything. Usually histogram have bars that represent frequency of occurring of data in the whole data set. A Histogram has two axis the x axis and the y axis. Here x axis represents the range of pixels whereas on the y axis it is the count of its intensities.

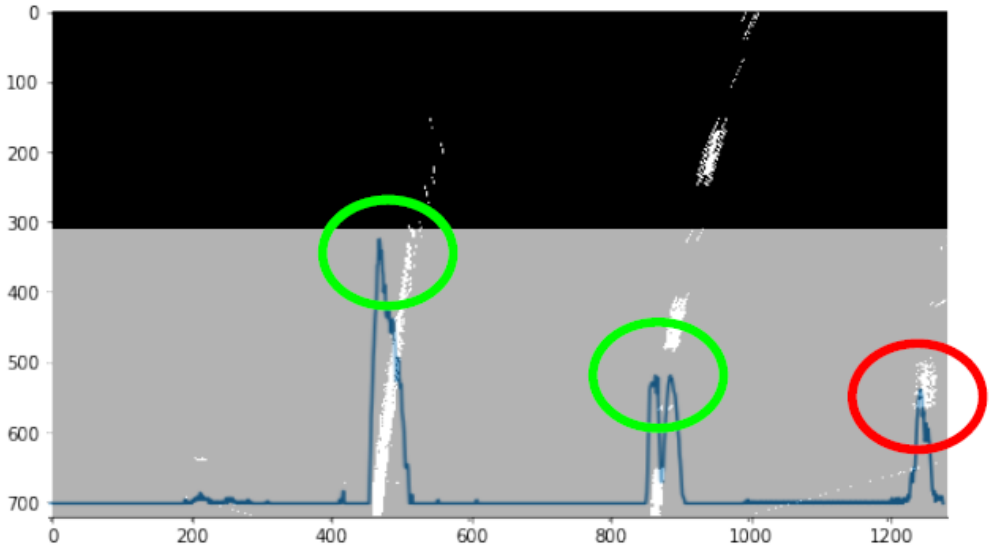
Histograms has many uses in image processing. The histograms has wide application in image brightness. Not only in brightness, but histograms are also used in adjusting contrast of an image. Another important use of histogram is to equalize an image.
And last but not the least, histogram has wide use in thresholding. This is mostly used in computer vision.

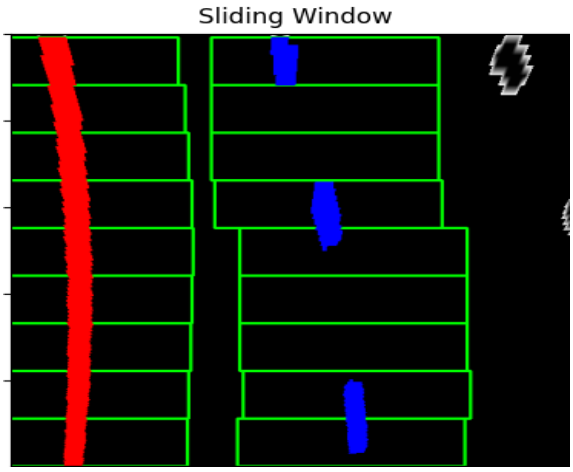
II. Sliding window method

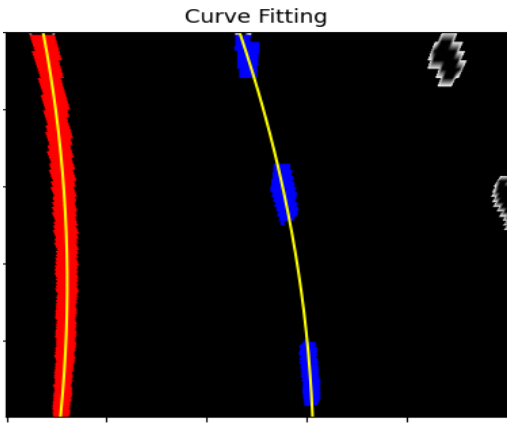
To verify the recognition performance of the lane detection algorithm in complex working conditions and dynamic environments, road driving videos were used to carry out the simulation test experiment in this work. Such videos in the experiment were collected by using a Carla simulator, and the lane lines were detected for different road conditions, such as highway, mountain road and tunnel road.

Furthermore, the classical sliding window search method was utilised to test the lane detection algorithm dynamically in the simulation test experiment. Firstly, a pixel histogram was generated by the pixels whose grey value was not equal to 0, and the peak value in the

pixel point set was obtained (the area where the lane line exists. Secondly, the sliding window was used to accumulate from bottom to top. When the number of pixels in the window exceeded the set threshold number, the mean value was taken as the centre of the next sliding window. In addition, by using the sliding window search method, the offset of the vehicle relative to the centre position of the road could be estimated.[4]





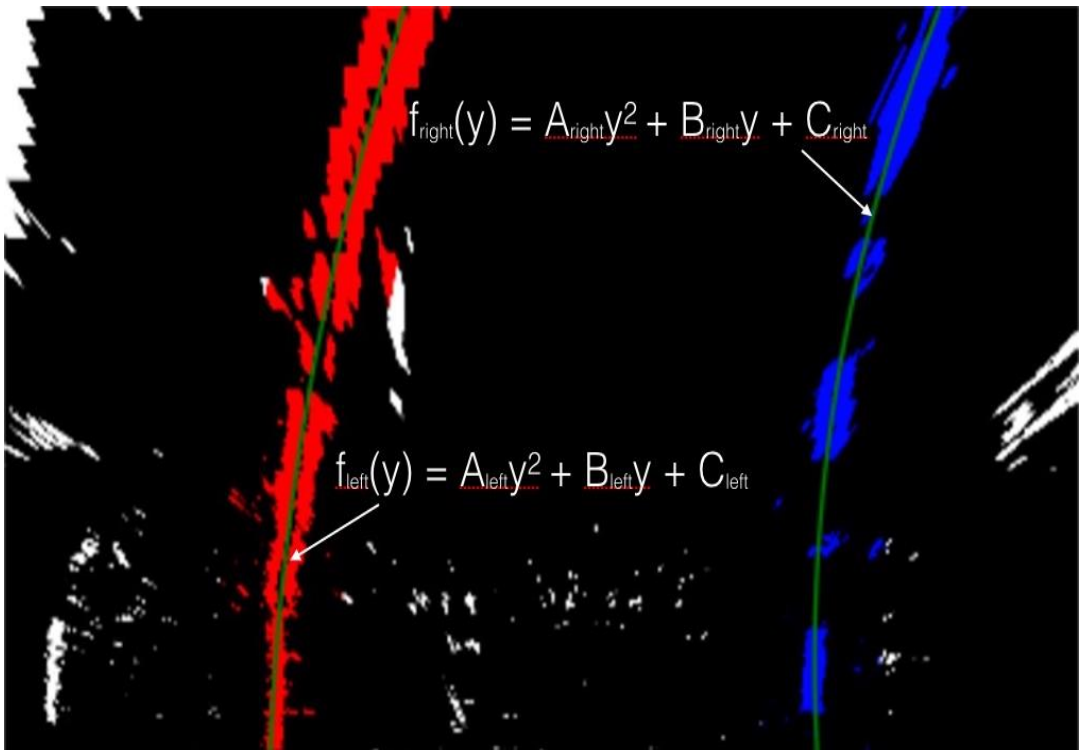


To recognize left and right lane line pixels, their *x* and *y* pixel positions are used, to fit a second-order polynomial curve:

$$f(y) = ay^2 + by + c = 0$$

The *f(y)* is used, rather than *f(x)* because the lane lines in the warped image are approximately vertical and may have the same x value for more than one y values.

III. Radius of curvature



The radius of curvature at any point x*x* of the function x = f(y)*x=f*(*y*) is given as follows:

$$R_{curve} = \frac{[1+(\frac{dx}{dy})^2]^{3/2}}{|\frac{d^2x}{dy^2}|}$$

In the case of the second order polynomial above, the first and second derivatives are:

$$f'(y) = \frac{dx}{dy} = 2Ay + B$$

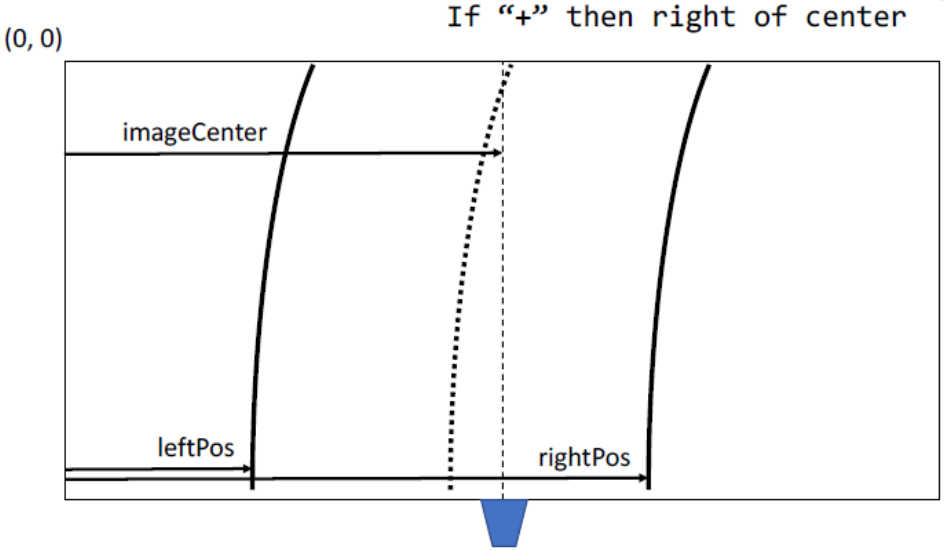
$$f''(y) = \frac{d^2x}{dy^2} = 2A$$

So, our equation for radius of curvature becomes:

$$R_{curve} = \frac{(1+(2Ay+B)^2)^{3/2}}{|2A|}$$

Distance to lane center

```
laneCenter = (rightPos - leftPos) / 2 + leftPos
distanceToCenter = laneCenter - imageCenter
If “+” then right of center
```



We've calculated the radius of curvature based on pixel values, so the radius we are reporting is in pixel space, which is not the same as real world space. So we actually need to repeat this calculation after converting our x and y values to real world space.

This involves measuring how long and wide the section of lane is that we're projecting in our warped image. We could do this in detail by measuring out the physical lane in the field of view of the camera, but for this project, you can assume that if you're projecting a section of lane similar to the images above, the lane is about 30 meters long and 3.7 meters wide. Or, if you prefer to derive a conversion from pixel space to world space in your own images, compare your images with U.S. regulations that require a minimum lane width of 12 feet or 3.7 meters, and the dashed lane lines are 10 feet or 3 meters long each.

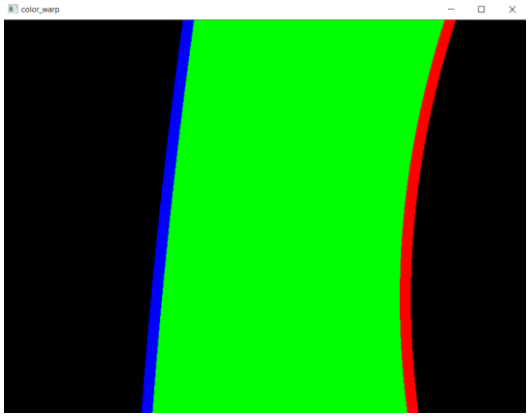
Let's say that our camera image has 720 relevant pixels in the y-dimension (remember, our image is perspective-transformed!), and we'll say roughly 700 relevant pixels in the x-dimension (our example of fake generated data above used from 200 pixels on the left to 900 on the right, or 700). Therefore, to convert from pixels to real-world meter measurements

Illustrate lines

Starting from the top point of view, the lane lines are easily recognized. A sliding window search distinguishes the lane lines. The green boxes express to the windows where the lane lines are colored. Since the windows search higher, they re-centre to the standard pixel position so that they resemble the lines.

The shaded lines will be stepped back onto the original image. The result of illustrated lanes from the sliding window search image is shown in Fig. 8.

After illustrating the lane lines, the warp and crop operations are performed for proper visualization of the image. To warp, a 3x3 transformation matrix operation performed. Straight lines will continue straight still after the transformation. To observe this transformation matrix, 4 points on the input image and the corresponding points on the output image are needed. Among those 4 points, 3 of them should not be collinear. Then the images are cropped because, during the determination of recognizing lane lines, we only need to focus on the regions where we are likely to see the lanes. The result of the warped and cropped image is presented in Fig. 9.



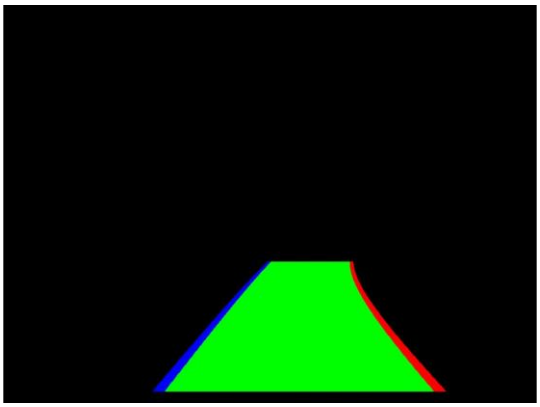


Fig8 (un-warp image) FIg 9 (warp image)

Conclusively, we get all this information and draw the results back onto the original image. The two pink lines which are recognized above are stated as lane lines, and the space between them is colored green to determine the road surface. While the pipeline prepares for a single image, it can easily be applied to processing many images to detect the lane line on the road surface. The final result of the proposed lane detection system is shown in Fig. 10.

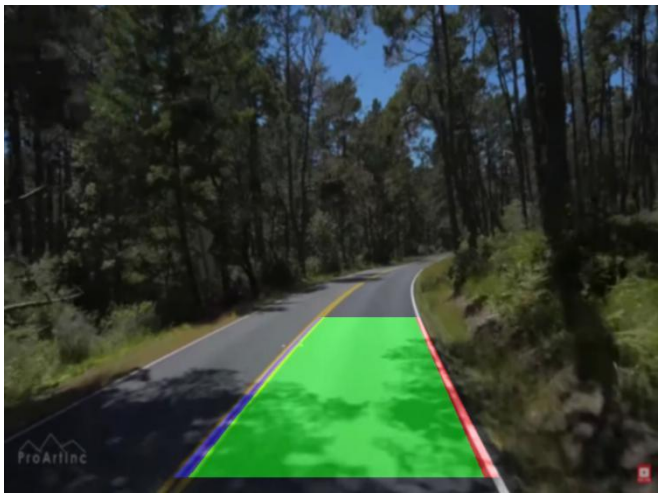
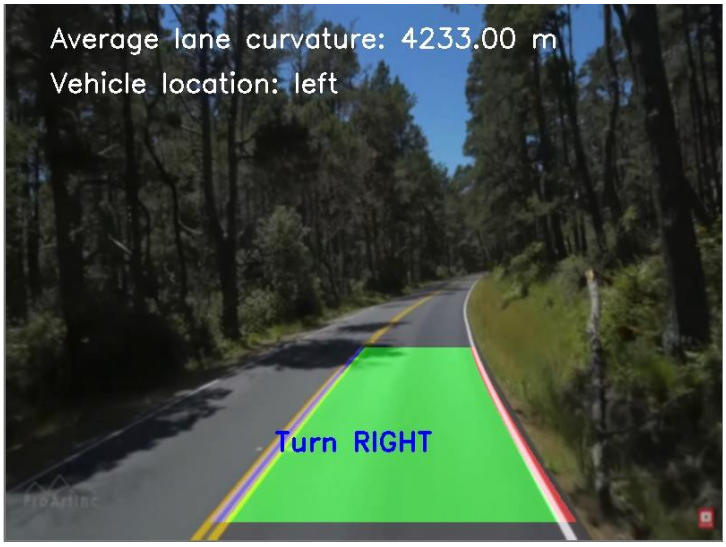


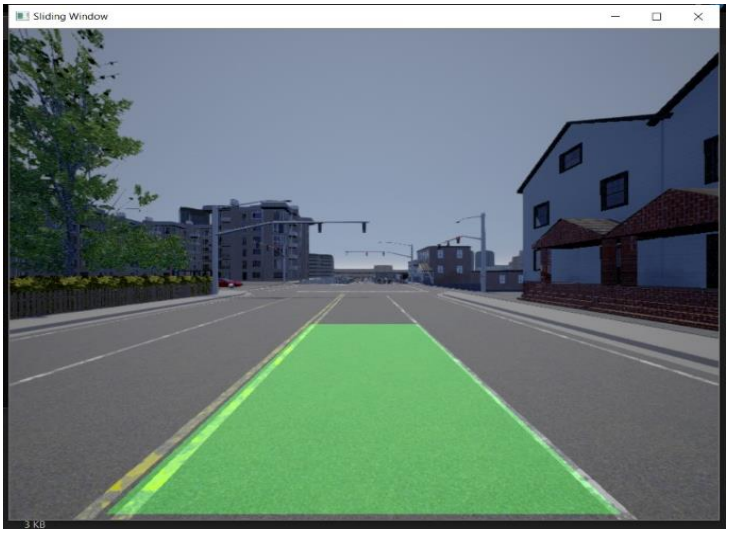
Fig10 (illustrate on the original image)

6. RESULT

Final output for lane detection on both, prerecorded video, and Carla.



Pre-recorded video



Carla

7. FUTURISTIC SCOPE

I. This project has various futuristic technologies which are still under R&D and hence it'll surely have use in the near future.

II. With an Industrial grade camera this system can become more robust.

8. CONCLUSION

The paper presents a lane detection system using computer vision-based technologies that can efficiently detect the lanes on the road. Different techniques like pre-processing, thresholding, perspective transform are fused together in the proposed lane detection system. Gradient and HLS thresholding detect the lane line in binary images efficiently. Sliding window search is used to recognize the left and right lane on the road. The cropping technique worked only the particular region that consists of the lane lines.

By using this computer vision algorithms we have tried to detect lane lines a simulated environment in real time with the help of open software like Carla which provide us a very robust system to implement our lane algorithm features into the simulated environment on a car.

From this we can conclude that the system detects the lanes efficiently with any conditions of the environment. The system can be applied to any road having well-marked lines and implemented to the embedded system for the assistance of Advanced Driver Assistance Systems and the visually impaired people for navigation to keep them in proper track.

Books

1. Digital Image Processing Book by Rafael C. GONZALES

REFERENCES

1. https://www.tutorialspoint.com/dip/perspective_transformation.htm
2. https://pechyonkin.me/portfolio/advanced-lane-finding/
3. https://jivp-eurasipjournals.springeropen.com/articles/10.1186/s13640-018-0326-2
4. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6679325/
5. http://proceedings.mlr.press/v78/dosovitskiy17a/dosovitskiy17a.pdf
6. https://www.udacity.com/course/self-driving-car-engineer-nanodegree--nd013
7. https://www.intmath.com/applications-differentiation/8-radius-curvature.php
8. https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_imgproc/py_thresholding/py_thresholding.html#thresholding