iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 8 **Issue:** V **Month of publication:** May 2020

DOI: http://doi.org/10.22214/ijraset.2020.5502

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429
Volume 8 Issue V May 2020- Available at www.ijraset.com

Analysis of Lane Detection Techniques on Structured Roads using OpenCV

Akash Punagin[1], Sahana Punagin[2]
[1]*Department of Computer Science, JSS SMI UG and PG Studies*

Abstract: ***The bone of contention for Advanced Driver Assistance Systems (ADAS) is Lane Detection. It is not effectively executed and is challenged because of susceptible computer vision techniques which are vulnerable for many factors like peculiar road pattern, unusual lighting and weather conditions, repercussion of obstacles like shadows and other cars, and many more. To try and overcome these vulnerabilities, The Sliding Windows method is used in this paper to detect lanes in real-time using the OpenCV python package. It includes undistorting, preprocessing, detecting lanes on the perspective transformed frame, and displaying detected lane as output. Radius of curvature of lane and offset of the car from the center of the lane is calculated. During preprocessing, different edge detection methods like Laplacian, Sobel and Canny edge detection methods are examined. After comparing and contemplating, it is deduced that Canny edge detection is better than the remaining ones. The test videos for lane detection are taken from Udacity GitHub repository.***
Keywords: Edge detection; kernel; sliding windows; lane detection; OpenCV

I. INTRODUCTION

Humans use optical perception to maneuver vehicles while driving. Approximately 1.35 million people die each year as a consequence of road traffic crashes [10]. Some of the common reasons for road accidents include speeding, driving under the influence of alcohol, non-use of motorcycle helmets, distracted driving, unsafe road infrastructure, unsafe vehicles, lack of post-crash care, and inadequate law enforcement of traffic rules [10]. As you can notice, humans are accountable for most of these reasons. We can count on machines, to avoid these happenings. An Autonomous vehicle is capable of perceiving its environment with little or no human input. Computer vision is a field of Artificial Intelligence that enables autonomous vehicles to understand the world around them. Autonomous Cars are required to perceive their surroundings. Based on their perceptions, they need to make decisions. The decisions taken from these perceptions are significant for executing actions like stopping at a red light, or stopping when a pedestrian walks by, or obeying the traffic signs, etc. There will be many sensors in an autonomous car for perception. This paper is fixated to the front-facing camera and analyzes the images taken from it to detect lanes. Lanes are essentially white or yellow markings on the road that intend to separate single lines of traffic according to speed or direction. They assist drivers and reduce traffic. Lane detection is closely related with Advanced Driver Assistance System (ADAS) and Driver Support System. While it seems easy to detect white or yellow markings on the road, it can be challenging because of unusual lighting or whether conditions, bizarre curves along the roads, occlusion by other vehicles etc.

This paper is divided into 10 sections. The second section gives the approach of the code. The third section gives information about the tools and libraries used. From fourth to eighth section, in order, deals with the steps involved in the algorithm. The results are discussed in ninth section. Ultimately, the paper is concluded in section ten.

II. APPROACH

The video or camera feed is read, frame by frame as input to the algorithm. In this paper, the test videos are taken from Udacity GitHub repository. Undistorting each frame will straighten the curved lines in the image which helps to identify the lane lines. Using Kernel Convolution, each frame is preprocessed by applying low-level filters. Kernel Convolution is used frequently in computer vision which is a process of passing a small grid of numbers over the whole image, transforming it based on the numbers in the grid. The grid is referred to as a ‘Kernel’. Different filters can be applied by using different numbers in the kernel. One of the filters is ‘Canny’, commonly known as the Canny edge detector. It intends to find optimal edges by applying thresholds which will lower the chance of detecting false edges. Applying Gaussian Blur before applying the Canny filter will also remove needless noise from the frame. After detecting edges, frames will be dilated and eroded to enhance the edges. Since lanes are usually white or yellow, a color filter is applied to accentuate the lane lines. Now, the preprocessing of frames concludes. The Sliding Window approach is applied to detect lanes and record points in which they are detected. These points are fit to a second degree polynomial from which Radius of curvature and Offset is calculated, for each frame. Block diagram of the approach is depicted in fig 1.

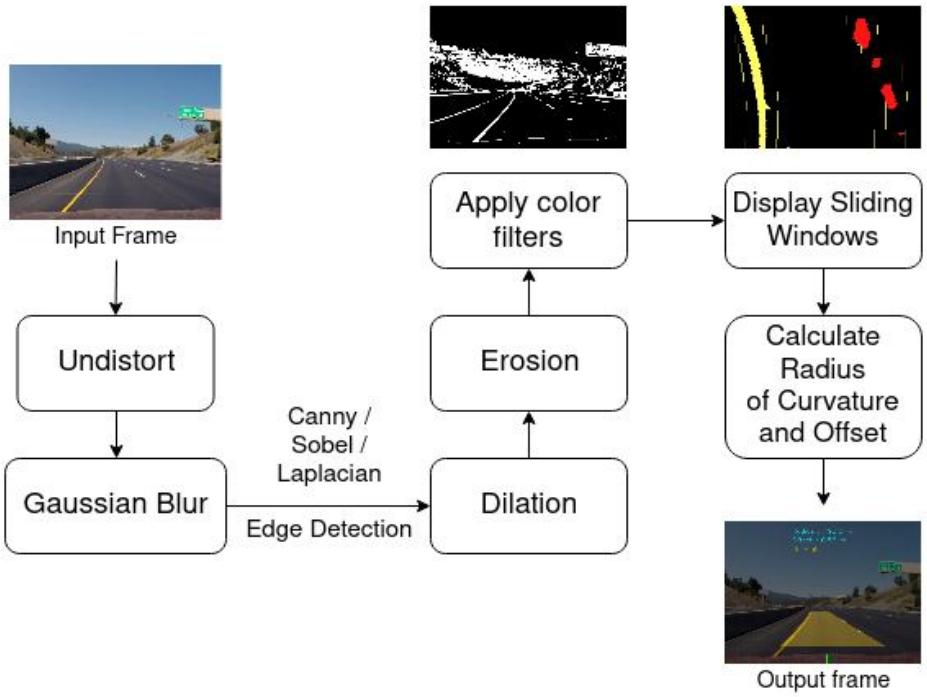


Fig. 1 Block diagram of Lane Detection Algorithm

III. TOOLS AND LIBRARIES USED

OpenCV (Open Source Computer Vision Library) is an open source library that includes hundreds of computer vision algorithms written in C and C++. It is mostly used for Artificial Intelligence, Machine Learning and Image Processing. It supports Windows, Linux, Mac OS, Android and iOS. It was designed to focus on real time computer vision applications.

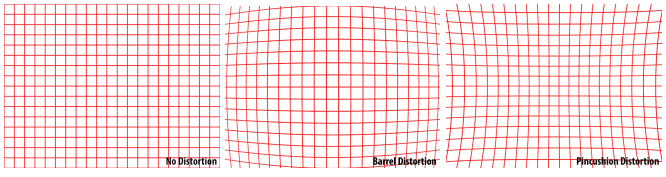
IV. TOOLS AND LIBRARIES USED

Distortion is an optical aberration that deforms and bends physically straight lines and makes them appear curvy in images. Distortion correction is a process of straitening the curved lines in an image. There are two types of Distortion; Radial and Tangential Distortion.

A. Radial Distortion

Radial distortion occurs when light rays bend more near the edges of a lens than they do at its optical center [1]. There are two types of Radial Distortion; Positive and Negative.

1) *Positive Radial Distortion (Barrel):* This is commonly seen in wide range lenses. It happens because the field of view of the lens is much wider than the size of the image sensor and hence it needs to be "squeezed" to fit. Hence, straight lines are curved inwards from the optical center. Fig 2 (b) illustrates Positive Radial Distortion [2]. Image sensor is an electronic device used in digital cameras to convert an optical image into an electronic signal to get a digital image. Straight lines are curved inwards from the optical center.
2) *Negative Radial Distortion (Pincushion):* This is commonly seen in telephoto lenses. It happens because the field of view of the lens is much smaller than the size of the image sensor and hence it needs to be "stretched" to fit. Hence, straight lines are curved outwards from the optical center. Fig 2 (c) illustrates Negative Radial Distortion.



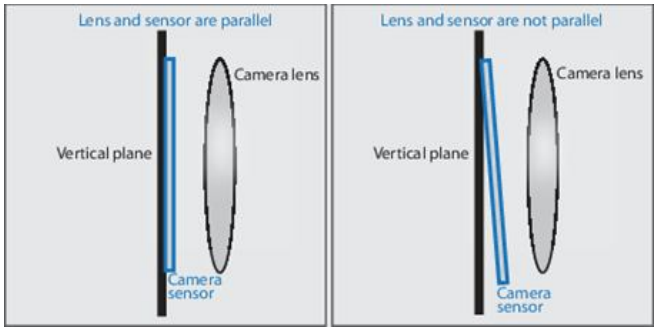
(a) (b) (c)

Fig. 2 (a) No Disortion, (b) Positive Radial Distortion, (c) Negative Radial Distortion

B. Tangential Distortion

Tangential distortion occurs when the lens and the image plane are not parallel [1].

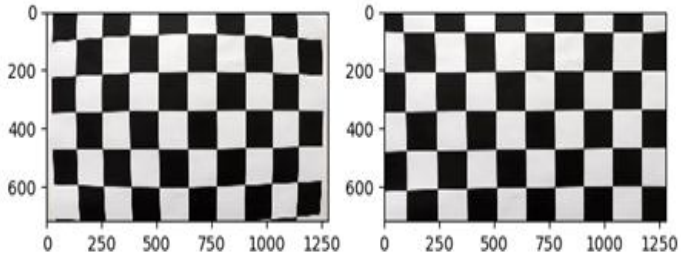
In fig 3(a), the Camera sensor is parallel to the lens and hence there is no Tangential Distortion. But in fig 3 (b), the camera sensor in tilted resulting in Tangential Distortion [1].



(a) (b)

Fig. 3 (a) Zero Tagential Distortion, (b) Tagential Distortion

Distortions can be corrected by calculating the Camera matrix and Distortion coefficients of distorted chess board images and mapping them to input frames. Distorted and Undistorted chess board images are shown in fig 4 (a) and (b) respectively



(a) (b)

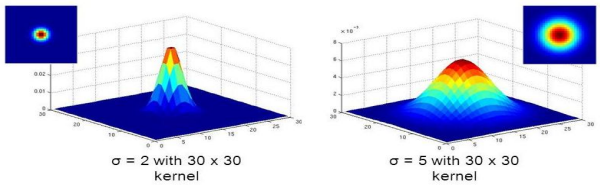
Fig. 4 (a) Distorted chess board, (b) Distorted chess board

V. PREPROCESSING FRAME

After undistorting the input frame, preprocessing is done to emphasize lane lines. It includes applying Gaussian Blur to smoothen the input frame. To detect edges, many edge detection methods are available. In this paper, Laplacian, Sobel and Canny edge detection methods are discussed.

A. Gaussian Blur

Before applying Gaussian blur, the frame should be converted to grayscale image. The pixels in that image are described as 2 dimensional array. The intensity of each pixel is represented by the values in that array. To blur or smoothen the image, the value of each pixel intensity should be replaced with the average value of pixel intensities around it. Averaging the pixel intensities will be done by a kernel called Gaussian kernel. To keep the energy stable, the values in the kernel usually adds up to 1. In Gaussian kernel, the values in the kernel are weighted based on normal distribution, to preserve the edges in the image. In fig 5 (a) and (b), 30 x 30 Gaussian kernel is visualized with Standard deviation of 2 and 5 respectively [3]. Extent of smoothing is determined by value of σ (Standard Deviation). 5 x 5 Gaussian kernel is shown in fig 6, the values in the kernel add up to 273. Therefore 273/273 will be 1. Each pixel in the input frame is multiplied by the Gaussian kernel. The Gaussian kernel is placed on the corner and moved across all pixels in the image. The pixels in each image that overlap the kernel will be multiplied by the corresponding kernel values. The values resulting from these multiplications will be added up and divided by the sum of kernel values to normalize the energy, to make sure the result does not get brighter or darker. The resulting value is stored in the destination pixel. The kernel needs to be flipped because of the mathematical properties of convolution. If the kernel is not flipped the result of convolution operation will be flipped. In fig 7, 3 x 3 Gaussian kernel is convolved with input frame. 3 rows and 3 columns of 2 dimensional array are involved in the calculation and are multiplied with respective kernel values (a,b,c...i). The resulting values are added and stored in the middle pixel in the output image.



(a) (b)

Fig. 5 30 x 30 Gaussian Kernel is visualized with Standard Deviation of (a) 2, (b) 5

$$\frac{1}{273}\begin{bmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{bmatrix}$$

Fig. 6 5 x 5 Gaussian Kernel

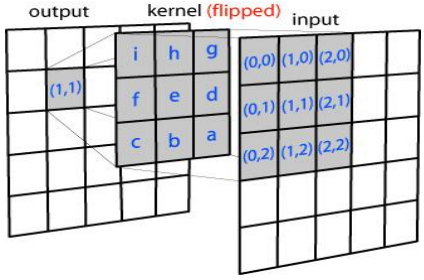


Fig. 7 3 x 3 Gaussian Kernel convolving with input frame

B. Edge Detection

Edge detection is a procedure of finding edges by detecting discontinuities in intensity or color in an image. A high value of implies a steep change and a low value implies a shallow change. By detecting edges, the data in the image will be significantly reduced and it preserves the structural properties for further image processing [4]. 3 types of edge detection methods are discussed below.

1) Sobel Edge Detection: Sobel operator was the most prevalent operator until the development of other edge detection operators with a better theoretical basis. It is a discrete differentiation operator which is a combination of Gaussian Blur (which reduces noise) and differentiation (which yields edge response) altogether [11]. Sobel operator calculates gradient change of an image in x and y direction with different kernels. In (1), G_x represents x direction kernel. Similarly, in (2), G_y represents y direction kernel. Calculating gradient change will give a set of values that represents the intensity of the vertical gradient and the horizontal gradient throughout the frame. Magnitude of vertical and horizontal gradients is calculated by (3).

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} \quad \dots (1)$$

$$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} \quad \dots (2)$$

$$G = \sqrt{G_x^2 + G_y^2} \quad \dots (3)$$

Applying Sobel's x direction kernel (G_x) to the input frame results in Horizontal Sobel Derivative (Sobel x) as shown in fig 8. Likewise, fig 9 represents Vertical Sobel Derivative (Sobel y) after applying y direction kernel (G_y).

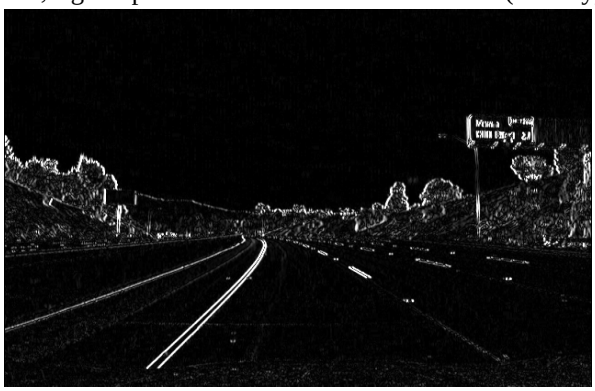


Fig. 8 Horizontal Sobel Derivative (Sobel x)

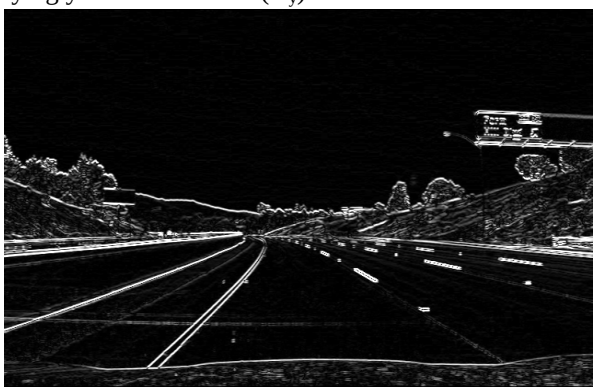


Fig. 9 Vertical Sobel Derivative (Sobel y)

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429
Volume 8 Issue V May 2020- Available at www.ijraset.com

2) *Laplacian Edge Detection*: Laplacian Operator is a gradient-based, derivative operator. But unlike other edge detection filters, which are first-order derivative filters (which detect edges based on local maxima or local minima), Laplacian operator is a second-order derivative filter (which detects edges at zero-crossing). It detects edges where the value switches from positive and negative and vice-versa. It produces uniform edge magnitude for all directions and gives better edge localization in comparison to first-order derivative filters. It calculates the 'Laplacian' of the image by adding second derivatives of x and y using Sobel operator only when the kernel size is greater than 1 as in (4) where, src represents the source frame and dst represents destination frame [12]. When kernel size is equal to 1, 'Laplacian' of an image is calculated by convolving with the kernel shown in (5).

$$dst = \Delta src = \frac{\partial^2 src}{\partial x^2} + \frac{\partial^2 src}{\partial y^2} \quad \ldots (4)$$

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad \ldots (5)$$

3) *Canny Edge Detection*: Canny edge detection is a multistage edge detector developed by John F. Canny in 1986. It is a multi-stage edge detector. It calculates the intensity of gradients by using a filter based on Gaussian edge detector. There are 4 stages in Canny edge detection [4]:

a) Remove noise in the image by convolving with Gaussian blur
b) Find the intensity gradients of the image
c) Apply non-maximum suppression to make the edges thin and remove noise around edges [5]. In fig 10, the blue line represents an edge. The yellow and pink line represents the edge neighbors If the gradient of the edge is larger than the gradient of edge neighbors, the edge is preserved.
d) Hysteresis thresholding: Detect and preserve dominant edges in an image [5]. In fig 11, Hysteresis thresholding is visualized in 1 dimension. The pink line indicates the edge response in an image. Two red arrows represent high and low threshold, they isolates pixels that has strong gradient. If the gradient is above the high threshold, it is automatically accepted as an edge. If the gradient is below the low threshold, it is automatically discounted as an edge. If the gradient is between the two thresholds, the gradient is preserved as an edge only of it is connected to any gradient above the high threshold.

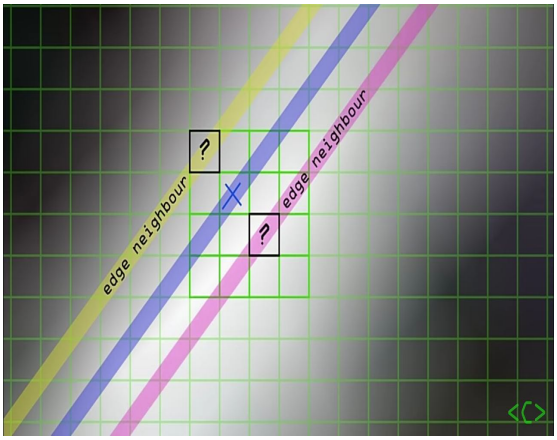


Fig. 10 Non-maximum suppression

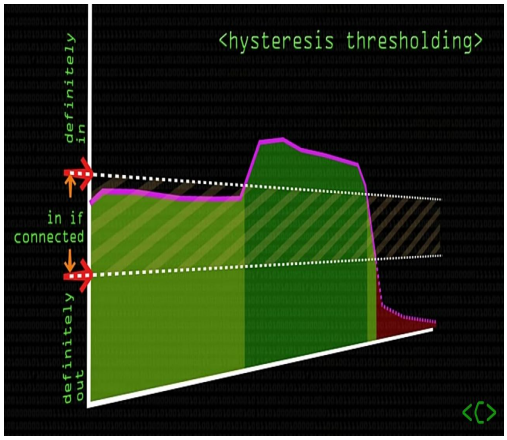
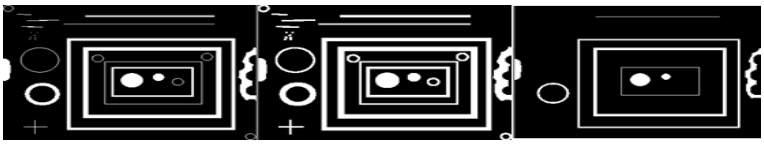


Fig. 11 Hysteresis thresholing

C. Dilation and Erosion

Dilation and Erosion are most basic Morphological Operations. Morphological Operations are image processing operations based on shapes [6].

1) Dilation adds pixels to the boundaries of foreground objects in the image. It accentuates features in an image. Fig 12 (a) and (b) represents original and dilated image respectively.
2) Erosion removes pixels from the boundaries of foreground objects in the image. It diminishes features in an image. Fig 12 (a) (c) represents original and eroded image respectively.



(a) (b) (c)

Fig. 12 (a) Original Image, (b) Dilated Image, (c) Eroded Image

Applying Color Filter: Lanes are usually yellow or white in color, therefore applying Yellow and White color filter will accentuate lane lines in an image. By applying color filter the lane lines can be clearly identified by the algorithm.

VI. APPLYING PERSPECTIVE TRANSFORM

If you stand between the railway tracks and look, they appear to be parallel near you. But if you look in the distance, they appear to converge at the horizon. This effect is called Perspective Effect [7]. This can be obliterated by Perspective Transformation. It gives a bird's-eye view of the image within source points. Four Source points can be drawn along the lanes to form a quadrangle as shown in the fig 13. The Bird's-eye view of the image within the Source points is shown in fig 14. Now, the lanes appear parallel (or close to parallel) to the camera and can be used for detection [7]. Since the perspective transformation is taking place only within the source points, only a sub-region of the image is transformed, resulting in reduced run time.

This is done programmatically in 2 steps:

1) Step 1: Apply the Perspective Transform Algorithm, & calculate Perspective Transform Matrix (M)

M = cv2.getPerspectiveTransform(src, dst)

Where, src is coordinates of quadrangle vertices in the source image and dst is coordinates of the corresponding quadrangle vertices in the destination image [8].

2) Step 2: Wrap the Transformed Image by applying Perspective Transform Matrix (M) to the image

warped_img = cv2.warpPerspective(frame, M, dst_size)

Where, frame is input image, M is 3 x 3 transformation matrix and dst_size is size of destination image [8].

Fig. 13 Source Points

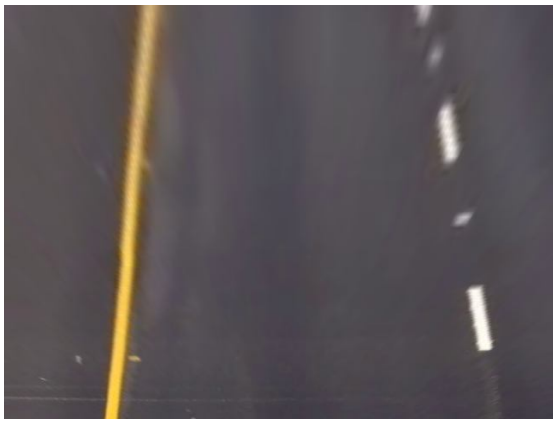


Fig. 14 Perspective wrapped frame (Bird's-eye view)

VII. DISPLAYING SLIDING WINDOWS

This is the main algorithm that does all the work of detecting lane lines in our image. The point in which lane lines are present in the image is a good initial point to get this algorithm started. Initial points are found by plotting a histogram of grayscale perspective wrapped frame along X-axis. The Histogram will illustrate the number of non-zero (white) pixels in individual columns of fig 15. The overlay of the image and histogram is shown in fig 16. The two peaks in the fig 16 represents the two detected lane lines in the image within the source points. Left and Right lane points can be differentiated by finding the midpoint along X-axis in the histogram, and finding the index of maximum element in both sides. To fit a curve along boundaries of the lane, we need all the points in the image in which the lanes are detected. So, staring from the initial points to the edge of the frame, the sliding window algorithm identifies all non-zero (white) pixels within each window and they are recorded. This saves a lot of processing time since we are detecting the lanes within the widows [9]. On each side, if the number of pixels detected in current window is more than the defined threshold of minimum pixels, then the initial points will be updated to the average of detected pixels on that side, and the next window will be shifted to the average lateral position of the detected pixels.

If the number of pixels detected in current window is less than the defined threshold of minimum pixels, the initial points will remain unaltered. The lanes detected by the algorithm will be identified by giving some defined color to them on either side. Fig 17 shows all the windows from initial points to the edge of the frame. Notice, on the right side, some windows are exactly on top of the previous window. This is because there are not sufficient pixels detected within the window to shift the next window laterally. Now we have all the points in which the lanes are detected on either side. Further, from these points we can fit a Second degree polynomial and obtain a curve that fit along the boundaries of the lanes by (6). Since the lane lines are approximately vertical, there might be same x values for two or more y values. Therefore, curve is fit along y-axis instead of x-axis [9].

$$f(y) = Ay^2 + By + C \quad \ldots (6)$$

At last we create an overlay that fills left and right points and combine the overlay with the frame. Fig 18 illustrates the overlay of lane on the input frame.

Fig. 15 Grayscale Perspective wrapped frame

Fig. 16 Overlay of histogram and input frame

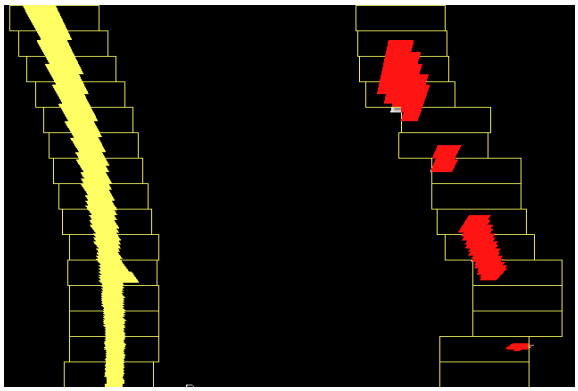


Fig. 17 Lanes detected by Sliding windows

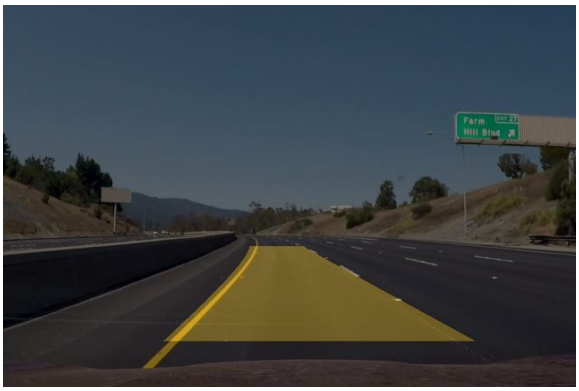


Fig. 18 Overlay of lane on input frame

VIII. CALCULATING RADIUS OF CURVATURE AND OFFSET

A. Calculating Radius of Curvature

It is essentially a measure expressing how much the lane is curved at a given point. In course of time, it can be used to control the speed of the car. It is depicted in fig 19.

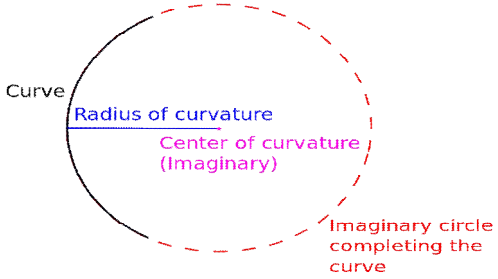


Fig. 19 Radius of Curvature

These measurements should be converted from pixel space to metric system. This information will be significant in the future, for creating programs that will control the steering and acceleration of the car. The formula for Radius of Curvature is given by (7), where the second order polynomial f(y) is defined by (8). The first and second derivatives of f(y) is given by (9) and (10) respectively. Substituting these in (7) will yield the formula that will be used in the program to calculate Radius of Curvature as specified in (11).

$$R_{curve} = \frac{\left[1 + \left(\frac{dx}{dy}\right)^2\right]^{\frac{3}{2}}}{\left|\frac{d^2x}{dy^2}\right|} \quad \ldots (7)$$

$$f(y) = Ay^2 + By + C \quad \ldots (8)$$

$$f'(y) = \frac{dx}{dy} = 2Ay + B \quad \ldots (9)$$

$$f''(y) = \frac{d^2x}{dy^2} = 2A \quad \ldots (10)$$

$$R_{curve} = \frac{[1 + (2Ay + B)^2)]^{\frac{3}{2}}}{|2A|} \quad \ldots (11)$$

B. Calculating Offset of the car

It is the measure of the position of the car relative to the lane. In this project, it is assumed that the camera is placed in the center of the car. From the left and right lane curves will determine where the lines hit the bottom the lane, adjacent to the car, by calculating its intercepts using second order polynomial f(y). Offset is calculated by averaging both the intercepts and converting it from pixel space to metric system.

IX. RESULTS

After the development of this code, different test videos were analyzed with all three edge detection methods as discussed. In fig 20, first column portrays all the outputs of Canny edge detector. The following columns portray Sobel and Laplacian edge detection outputs. Row 1 shows detected edges in the frames. The preprocessed frames are represented in Row 2. These frames are perspective wrapped to obtain a bird's-eye view which is depicted in Row 3. Sliding Windows algorithm is applied for each frame to detect lane points and is shown in Row 4. Using these points, left and right curves are calculated. An overlay of a defined color is formed between the left and right curves. Radius of Curvature and Offset are calculated. The detected lanes and with the calculations are shown in Row 5.

Therefore, we can observe that, the results obtained are analogous to each other. The lanes are imperfectly detected when Sobel and Laplacian edge detectors are used, unlike Canny edge detector. It is also reflected in all the steps above. Sobel and Laplacian edge detectors cannot produce thin and smooth edges. Canny edge detection method suppresses noise and removes unwanted textures using non-maximum suppression.

X. CONCLUSION

This paper proposes a robust lane detection algorithm using Sliding Windows approach which involves undistorting and detecting edges using three different edge detection methods: Canny, Sobel and Laplacian edge detection. After contemplating the results, it is understood that Canny edge detection is the most efficient method for edge detection as compared to Sobel and Laplacian edge detection.

Although, this paper detects lanes on structured roads, it can be used to detect lanes on unstructured roads also. But there will be complications like detecting edges and lane lines. In some situations, the light might be inadequate too luminous. To overcome this, thresholds of color filter and canny edge detectors can be made adaptive.

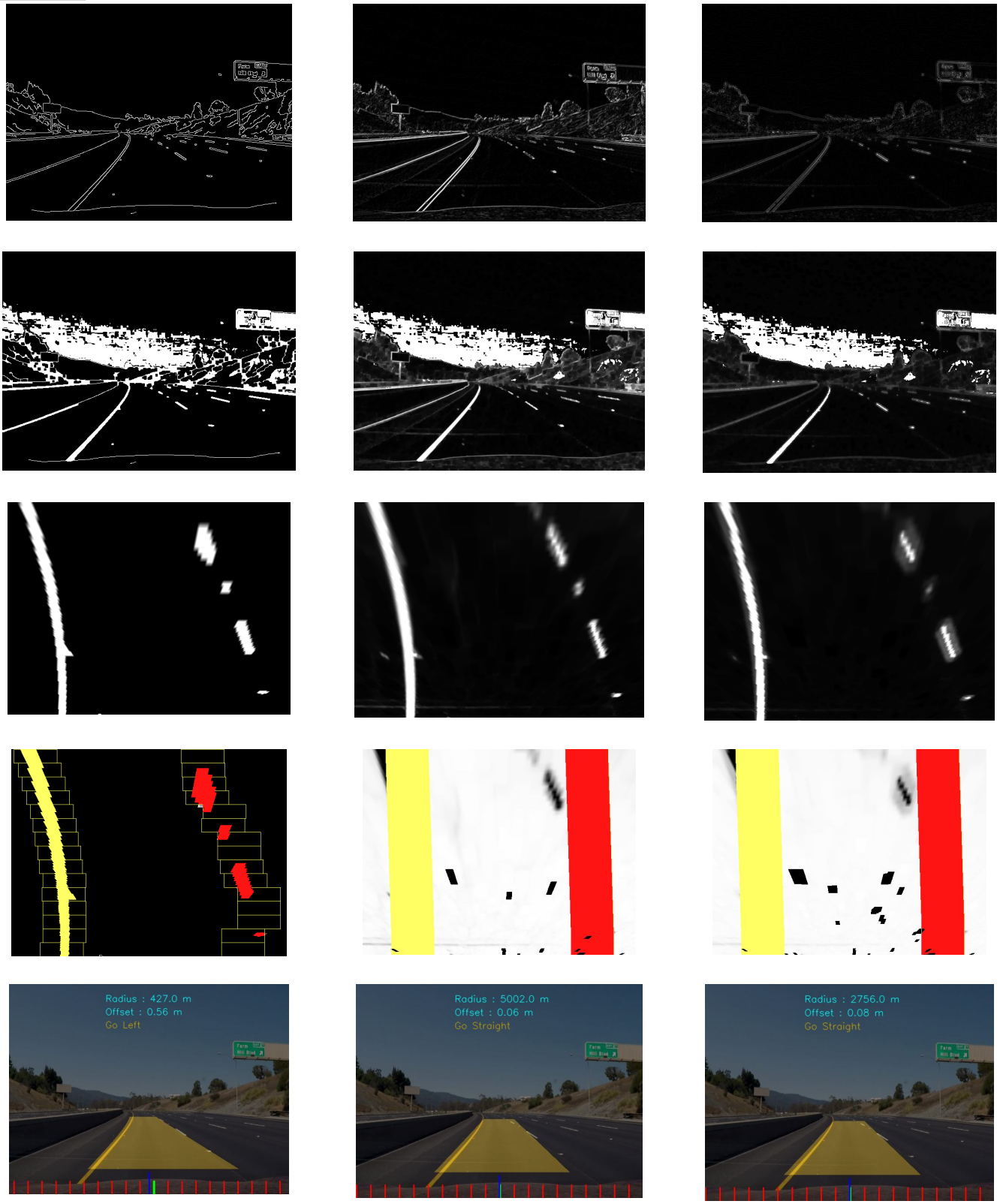
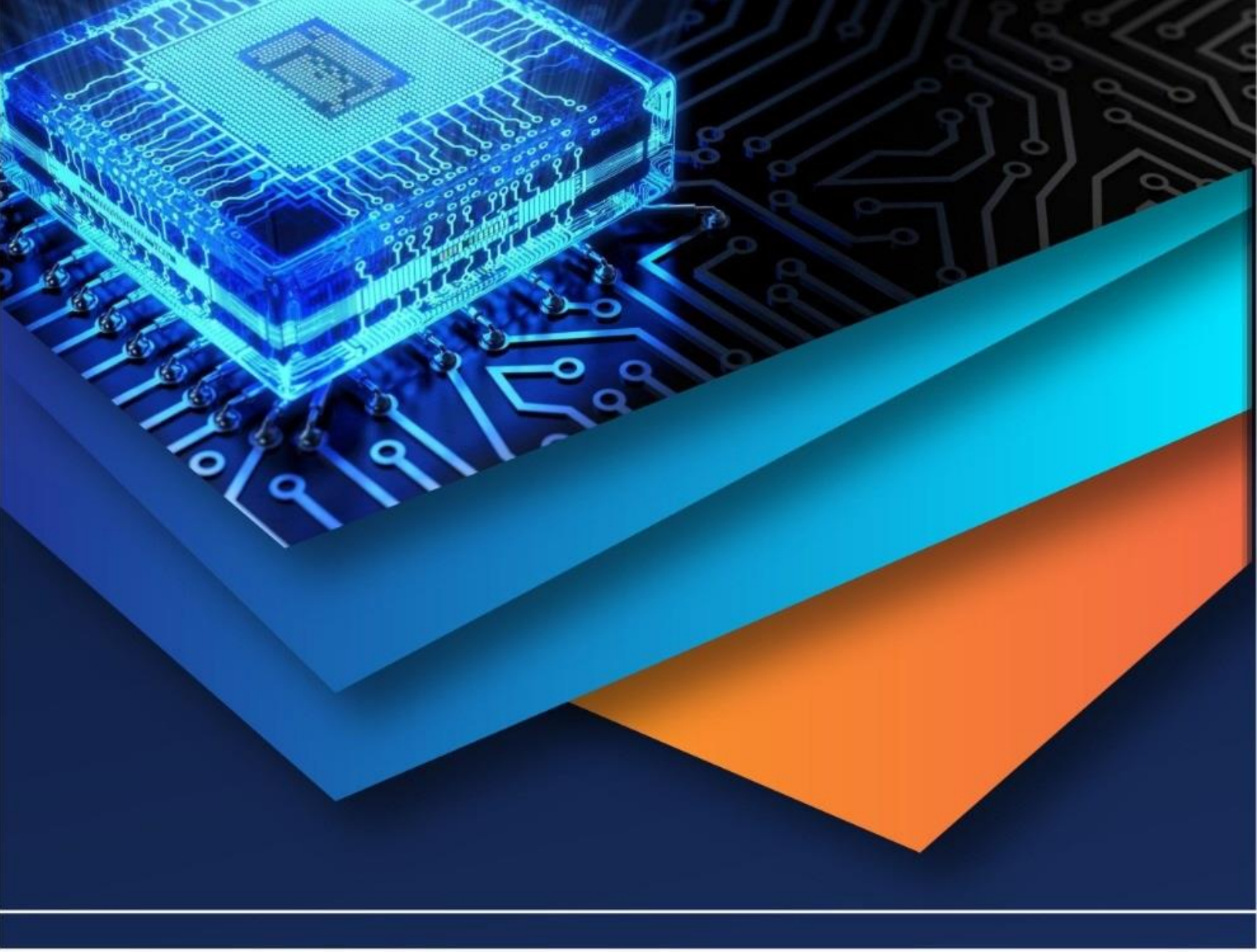


Fig. 19 Final results describing each step for Canny, Sobel and Laplacian Edge Detection methods (column wise). Row 1 : Edges detected, Row 2 : Preprocessed Frame, Row 3 : Perspective wrapped frame, Row 4 : Frames with Sliding Windows, Row 5 : Detected Lanes with calculations

REFERENCES

[1] https://www.mathworks.com/help/vision/ug/camera-calibration.html
[2] https://photographylife.com/what-is-distortion
[3] K. Grauman, “Gaussian Kernal”
[4] Rashmi, Mukesh Kumar and Rohini Saxena “ALGORITHM AND TECHNIQUE ON VARIOUS EDGE DETECTION: A SURVEY” Signal & Image Processing : An International Journal (SIPIJ) Vol.4, No.3, June 2013 DOI : 10.5121/sipij.2013.4306
[5] Computerphile, YouTube channel
[6] https://www.mathworks.com/help/images/morphological-dilation-and-erosion.html
[7] Mohamed Aly “Real time Detection of Lane Markers in Urban Streets” Computational Vision Lab, Electrical Engineering, California Institute of Technology, arXiv:1411.7113v1 [cs.CV] 26 Nov 2014
[8] https://docs.opencv.org/2.4/modules/imgproc/doc/geometric_transformations.html
[9] Md. Rezwanul Haque, Md. Milon Islam, Kazi Saeed Alam, Hasib Iqbal, Md. Ebrahim Shaik, " A Computer Vision based Lane Detection Approach", International Journal of Image, Graphics and Signal Processing(IJIGSP), Vol.11, No.3, pp. 27-34, 2019.DOI: 10.5815/ijigsp.2019.03.04
[10] https://www.who.int/news-room/fact-sheets/detail/road-traffic-injuries
[11] https://docs.opencv.org/3.4/d2/d2c/tutorial_sobel_derivatives.html
[12] https://docs.opencv.org/3.4/d4/d86/group__imgproc__filter.html#gad78703e4c8fe703d479c1860d76429e6



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY