International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 07 Issue: 08 | August - 2023 SJIF Rating: 8.176 ISSN: 2582-3930

Enhanced Lane Detection Techniques on Structured Roads using OpenCV

Akshay Mulgund [1], Prof. Shivakumara T [2]

[1]*Post Graduate Student, Department of Master of Computer Application, BMS Institute of Technology and Management, Bengaluru, Karnataka, India*

[2]*Assistant Professor, Department of Master of Computer Application, BMS Institute of Technology and Management, Bengaluru, Karnataka, India*

Abstract— Lane detection remains a significant challenge in Advanced Driver Assistance Systems (ADAS) due to its susceptibility to various factors such as complex road patterns, adverse lighting and weather conditions, and the presence of obstacles like shadows and other vehicles. To address these vulnerabilities, this paper proposes a real-time lane detection approach utilizing the Sliding Windows method and the Canny edge detection technique implemented with the OpenCV Python package. The proposed method includes distortion correction, preprocessing, lane detection on perspective-transformed frames, and lane visualization as output. Additionally, the method calculates the lane's radius of curvature and the car's offset from the lane center. In the preprocessing stage, several edge detection methods, including Laplacian, Sobel, and Canny, are evaluated. Through rigorous comparison and analysis, the superiority of the Canny edge detection method is demonstrated. The performance of the proposed approach is evaluated on test videos obtained from the Udacity GitHub repository, showcasing its effectiveness in challenging scenarios.

Index Terms: openCV,Canny

1. INTRODUCTION

The proliferation of autonomous vehicles presents a promising avenue to address the alarming global toll of road traffic crashes, wherein human negligence is often a primary catalyst. The deployment of self-driving cars, empowered by computer vision techniques, holds the potential to significantly enhance road safety by mitigating the impact of common causes of accidents, such as speeding, driving under the influence of alcohol, and distracted driving.

This paper is devoted to the exploration of an essential aspect of autonomous driving - lane detection. Lanes, demarcated by white or yellow markings, serve the fundamental purpose of segregating traffic flows based on speed and direction. Besides enhancing traffic organization, they play a crucial role in assisting drivers in maintaining their course. However, detecting these markings reliably can be a daunting task due to challenging scenarios, including varying lighting conditions, adverse weather, complex road geometries, and occlusion caused by surrounding vehicles.

The primary objective of this research is to propose an effective lane detection algorithm that utilizes data captured from front-facing cameras on autonomous vehicles. These cameras serve as a key sensor in the perception system, enabling the vehicle to comprehend its environment and make crucial decisions, such as stopping at red lights or yielding to pedestrians.

The proposed algorithm will be an integral part of an Advanced Driver Assistance System (ADAS), acting as a crucial component of driver support. By accurately identifying and tracking lanes in real- time, the ADAS can provide timely warnings, assist with lane-keeping, and facilitate safe navigation through complex road networks.

This paper is structured into ten sections. The second section presents an overview of the lane detection

approach. In the third section, the tools and libraries employed in the development of the algorithm are described. Subsequently, sections four to eight delve into the detailed steps of the lane detection algorithm. The ninth section discusses the results obtained from the algorithm's evaluation, and finally, the paper concludes in section ten, summarizing the findings and highlighting future directions for research in this critical area of autonomous vehicle technology.

2. APPORACH

The video or camera stream is read frame by frame by the algorithm as input. The Udacity GitHub source was used to obtain the test movies for this study. By undistorting each picture, the wavy lines in the image will become straight, making it simpler to see the Lane markers. Low-level filters are used to preprocess each frame using Kernel Convolution. Kernel Convolution is employed.In computer vision, a process of translating a picture based on the numbers in the grid, a small grid of numbers is frequently utilized. The grid is referred to as a "Kernel". Various filters can be applied using different integers in the core. The "Canny" filter, also referred to as the "Canny edge detector," is one of the filters. When applied before the Canny filter, Gaussian Blur will also remove excess noise from the frame.

When edges are found, frames will be enlarged and degraded to highlight the edges. Lane markings are commonly white or yellow, therefore they are highlighted with a color filter. The frames have now finished being prepared. Lanes are recognized and their positions are marked using the Sliding Window approach. The Radius of Curvature and Offset of each frame are computed from the fitted points to a second-degree polynomial. Figure 1 shows a block diagram of the procedure.

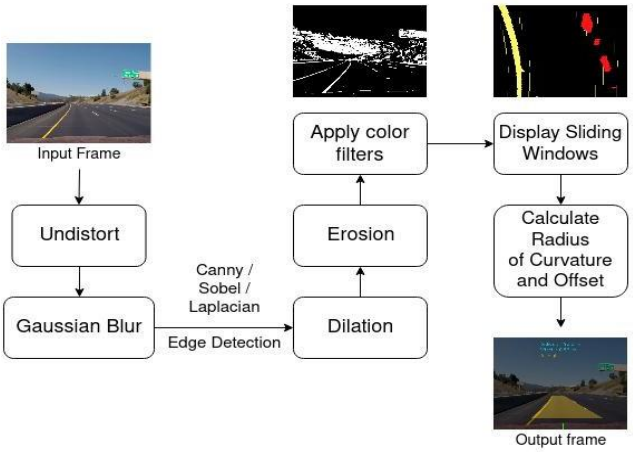


Figure 1 Block diagram of Lane Detection Algorithm

3. TOOLS AND LIBRARIES USED

A range of tools and libraries are used to perform lane detection in the context of autonomous vehicles and advanced driver support systems. For computer vision applications like edge detection and contour analysis, the OpenCV library is frequently utilized. NumPy makes it easier to manage picture data effectively, while Scikit-learn makes it possible to incorporate machine learning methods as needed. Visualizing picture data and findings is made easier with Matplotlib. The fundamental techniques frequently use Gaussian blur for noise reduction, the Hough Transform for line detection, and Canny edge detection. Using Region of Interest (ROI) masking, significant regions of the image are separated, and Real-time video is recorded using computer vision cameras. Convolutional neural networks (CNNs) can be used with lane segmentation using deep learning frameworks like TensorFlow or PyTorch for more sophisticated applications. Furthermore, ROS offers a platform for robotics systems to incorporate lane detecting techniques. To guarantee safety and protect people's privacy when driving, it's essential to adhere to ethical norms and give credit where credit is due.

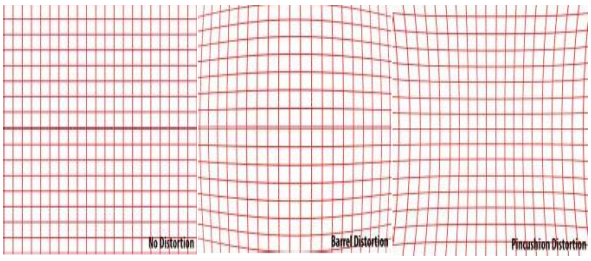
4. METHODOLOGY

Because of distortion, an optical aberration, which bends and deforms physically straight lines, they look curved in images. In distortion correction, curved lines are straightened in the image. The two different types of distortion are tangential and radial distortion.circular distortion.Radial distortion happens when light rays bend differently near a lens's edges than they do at its optical center [1]. There are two types of radial distortion: positive and negative.

1) Positive radial distortion: (barrel) is a common symptom of wide-angle lenses. It happens when the lens' field of view is far wider than the size of the image sensor, forcing the lens to be "squeezed" to fit. Straight lines are curled as a result.internal to the optical center. Fig. 2 (b) [2] shows positive radial distortion. An electrical component known as an image sensor is used in Digital cameras convert an optical image into an electrical signal to produce a digital image. Straight lines are curved inward.the optical center, I suppose.

2) Negative radial distortion (Pincushion): which is typically seen in telephoto lenses. It occurs because the lens' field of view is much smaller than the size of

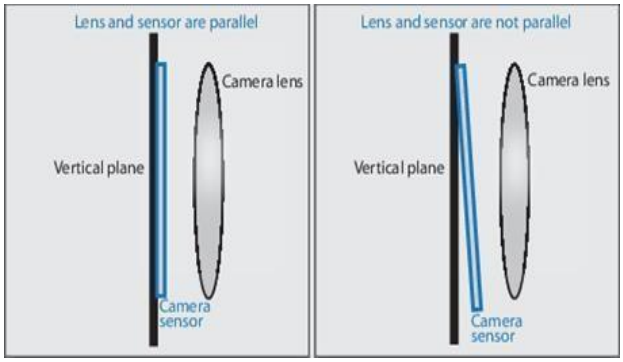
the image sensor and must be "stretched" to fit. Straight lines are consequently bent away from the optical center. In Fig. 2(c), a negative radial distortion is depicted.



(a) (b) (c)

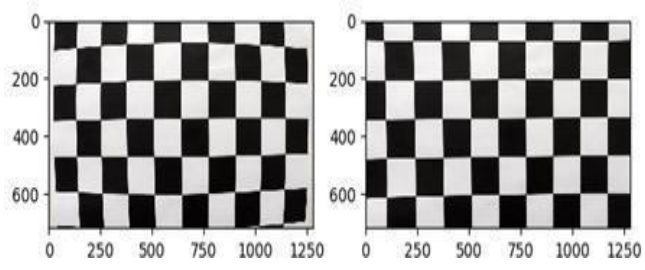
B. Tangential Distortion

When the lens and the image plane are not parallel, tangential distortion happens [1].Since the camera sensor and lens are parallel in Fig. 3(a), there is no tangential distortion. However, the camera sensor is slanted in Fig. 3(b), which causes Tangential Distortion [1].



(a) (b)

Figure 3 (a) Zero Tagential (b) Tagential Distortion



(a) (b)

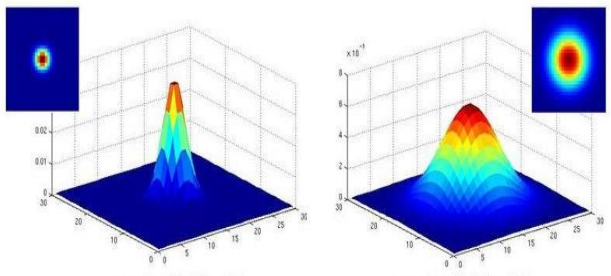
Figure 4 (a) Distorted chess board, (b) Distorted chess board

4. PREPROCESSING FRAME

After the input frame has been corrected for distortion, lane lines are highlighted during preprocessing. Gaussian Blur is also used to blur the input frame. There are many edge detection methods out there. This article discusses the Laplacian, Sobel, and Canny edge detection techniques.

1) Before applying Gaussian blur, the frame should be converted to a grayscale image. A two-dimensional array is used to describe the pixels in that image. The intensity of each pixel is represented by the values in that array. To Instead of using the value of each pixel, the image should be smoothed using the average value of the intensities of the pixels around it.The pixel intensities will be averaged using a kernel called the Gaussian kernel. The values in the Kernel must add up to 1 in order to maintain constant energy. Based on the normal, the values in the Gaussian kernel are weighted.

2) Distribution, to keep the edges of the image intact.The values of the Gaussian kernel are weighted according to the normal distribution to preserve the edges of the image. Figures 5(a) and (b) depict the 30 x 30 Gaussian kernel, which has standard deviations of 2 and 5, respectively [3]. The amount of smoothing is determined by the Standard Deviation number. A 5 x 5 Gaussian kernel with values totaling 273 is shown in Therefore, 273/273 will equal 1. Each pixel in the input frame is multiplied by the Gaussian kernel. 2) The image's Gaussian kernel is shifted from a corner position and moved across each pixel. For each image's kernel-overlapping pixels, the corresponding kernel will be present. multiplying values by them. the values produced.Convolution's effects will be reversed. In Fig. 7, a 3 x 3 Gaussian kernel is used to convolve the input frame. A two-dimensional array with three rows and three columns is multiplied in the calculation.The correct kernel values, such as a, b, and c.



σ = 2 with 30 x 30 kernel (a) σ = 5 with 30 x 30 kernel (b)

Figure 5 30 x 30 Gaussian Kernel is visualized with Standard

Figure 5.1 5 x 5 Gaussian Kernel

2)Edge Detection: Image 5.1, 5 x 5. Edge detection, also known as Gaussian Kernel 2, is a method for recognizing edges in a photograph by spotting variations in color or intensity. A sudden change is inferred when is high, and a gradual transition is implied when is low. Edge detection will significantly minimize the amount of data in the image while preserving the structural characteristics for further image processing [4]. Three distinct edge detection strategies are presented below.

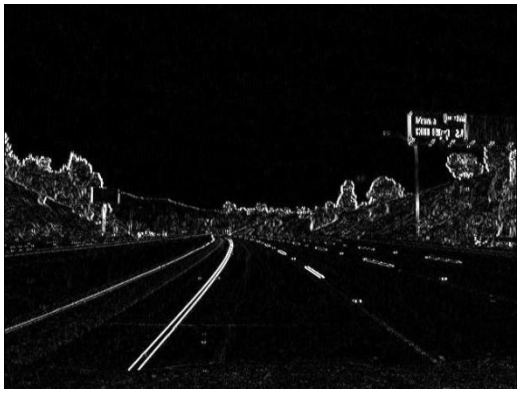
a)Sobel Edge Detection:Picture 5.1, 5 x 5. By identifying differences in hue or intensity, edge detection, also known as Gaussian Kernel 2, is a technique for identifying edges in photographs. When is high, a quick change is conveyed; when it is low, a gradual change is suggested. The amount of data in the image will be greatly reduced by edge detection but the structural properties are kept for additional image processing [4]. Below are three different edge detecting methods.

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

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

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

The Horizontal Sobel Derivative (Sobel x) is produced by applying Sobel's x direction kernel (Gx) to the input



frame, as shown in figure 5.3. Similar to shows the

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

Vertical Sobel Derivative (Sobel y) following application of the y direction kernel (Gy).

Figure 5.2 Horizontal Sobel Derivative (Sobel x)

b)Laplacian Edge Detection: A gradient-based derivative operator is the Laplacian Operator. Contrary to other edge detection filters, which are first-order derivative filters that identify edges based on local maxima or local minima, the Laplacian operator is a second-order derivative filter that identifies edges at zero-crossing. It detects boundaries when a positive value turns negative or vice versa. It generates different results than first-order derivative filters.

Enhanced edge localization and constant edge magnitude in all directions. The "Laplacian" of the image is determined by summing the second derivatives of x and y using the Sobel operator [12] when the kernel size is greater than 1, as in equation (4). Src and dst stand for the source and destination frames, respectively.

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

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

6. APPLYING PERSPECTIVE TRANSFORM

If you stand between the train tracks and look in that direction, they appear parallel to you. If you look far away, they appear to converge close to the horizon. This effect is known as the perspective effect [7]. This is something that perspective can eliminate. Transformation. It gives the image seen within the source points an aerial perspective. Four source locations can be identified along the roads. to form a quadrilateral similar

to the one in figure 13. Figure 14 shows the image contained within the Source points from an aerial perspective. Now, it is possible to use lanes that seem parallel to the camera or almost parallel to it for detection [7].

This is done programmatically in 2 steps:

Step 1: Calculate the Perspective Transform Matrix (M) by applying the Perspective Transform Algorithm and setting M = cv2.(src, dst) get Perspective Transform where dst is the coordinates of the corresponding quadrangle vertices in the destination image and src is the coordinates of the quadrangle vertices in the source image.

Step 2: Apply the Perspective Transform Matrix (M) on the image warped_img = cv2.warp to wrap the transformed picture.Frame is the input image, M is a 3 x 3 transformation matrix, and dst_size is the size.

Figure 6.1 Source Points

7. DISPLAYING SLIDING WINDOWS

This is the main algorithm that finds all the lane lines in our image. Where the lane lines initially emerge in the image is a great place to start using this technique. Initial points are produced by plotting a grayscale perspective wrapped frame's histogram down the X-axis. A histogram will display the percentage of non-zero (white) values. each of the 15 columns contains pixels. The image and histogram are juxtaposed in Figure 16. The two peaks in represent the two lane lines that were discovered inside the image's source regions. By identifying the histogram's center on the X-axis and calculating .To fit a curve along the lanes' limits, we need to know the location of each point in the image where lanes can be found. Starting from the initial points to the frame's edge, the sliding window technique detects and records all non-zero (white) pixels within each window. This considerably speeds up processing because we are finding the lanes inside the widows [9]. If the number of pixels detected in the current window exceeds the specified threshold of minimum pixels on each side, the initial points will be updated to the average of the detected pixels on each side, and the next window will be relocated to the average lateral position of the detected pixels.

The lanes on either side will be identified by the computer by being assigned a specific color. Each window is shown in Figure 17 from its starting positions to the frame's boundary. Some of the windows on the right side exactly overlap the one before them, as you'll see. The window's lack of pixels is to blame for this. could be discovered to laterally move the following window. The entire area that has lanes on either side has been covered. Additionally, we can fit a second-degree polynomial to these sites using the formula (6) to create curve lane borders. Two could have identical x values since the lane lines are nearly vertical.Therefore, the curve is fitted along the y-axis rather than the x-axis.

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

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

Figure 7.1 Overlay of lane input

8. CALCULATING RADIUS OF CURVATURE AND OFFSET

A.Calculating Radius of Curvature:In essence, it is a measurement that indicates how far the lane curves at a specific place. Over time, it may be employed to regulate the vehicle's speed.It is necessary to convert these measurements from pixel space to the metric system.

Future programming that regulates the steering and acceleration of the car will greatly benefit from this knowledge. The second order polynomial f(y) is defined by, while the formula for Radius of Curvature is provided by. Given by and, respectively, are f(y)'s first and second derivatives. The formula that will be utilized in the software to compute the radius of curvature as stated in will be produced by substituting these into the equation.It is a measurement of the car's position in relation to the lane.

The camera will be positioned in the middle of the vehicle for the purposes of this project. By calculating the intercepts of the lines using the second order polynomial f(y), it is possible to ascertain where the lines intersect the bottom of the lane, next to the automobile, based on the left and right lane curves. By averaging both intercepts and changing the data from pixel space to metric system, offset is determined.

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

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

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

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

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

B.Calculating Offset of the car:

It is a measurement of the car's position in relation to the lane. The camera will be positioned in the middle of the vehicle for the purposes of this project. By calculating the intercepts of the lines using the second order polynomial f(y), it is possible to ascertain where the lines intersect the bottom of the lane, next to the automobile, based on the left and right lane curves. By averaging both intercepts and changing the data from pixel space to metric system, offset is determined.

9. RESULTS

Following the creation of this code, various test videos were examined using all three of the previously mentioned edge detection techniques.The columns below display the outcomes of edge detection using Sobel and Laplacian.displays the frames' edges that were located. The preprocessed frames are displayed in Row. These frames were perspective wrapped to get the aerial viewpoint shown in Row 3. The lanes are located for each frame using the Sliding Windows method, as demonstrated in Row 4. The left and right curves are computed using these points. The intersection of the left and right curves results in a distinct color overlay. Offset and Radius of Curvature are calculated. Both calculated and detected lanes are shown in the display. As a result, we are able to see that the results are comparable. When it comes to detecting the lanes, Sobel and Laplacian edge detectors perform worse than Canny edge detector. Each of the aforementioned procedures exhibits it as well. Sobel and Laplacian edge detectors are unable to produce thin and smooth edges. The Canny edge detection method minimizes noise and gets rid of unwanted textures via non-maximum suppression.

10. CONCLUSION

The Canny, Sobel, and Laplacian edge detection methods are used to undistort and identify edges using the Sliding Windows approach, which is proposed in this paper. The results show that Canny edge detection is the most efficient edge detection method, outperforming Sobel and Laplacian edge detection.

This paper focuses on organized roadways. It can also be applied for unstructured roads to find lanes. However, there will be difficulties, such as lane markings and edge detection. The light may be excessively bright or insufficient in some circumstances. To get around this, adjustable thresholds can be built for color filters and clever edge detectors.

REFERENCES

1. Canny, J. (1986). A computational approach to edge detection. IEEE Transactions on Pattern Analysis and Machine Intelligence, 8(6), 679-698.

2. Sobel, I., & Feldman, G. (1968). A 3x3 Isotropic Gradient Operator for Image Processing. Stanford Artificial Intelligence Project, Memo AIM-52.

3. Hough, P. V. C. (1962). Method and Means for Recognizing Complex Patterns. U.S. Patent No.3,069,654.

4. Shi, J., & Tomasi, C. (1994). Good Features to Track. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 593-600).

5. Son, J., Kim, J., & Heo, Y. (2016). Robust lane detection and tracking in challenging scenarios.In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW) (pp. 47-55).

6. Lee, H. G., Eom, J. H., & Yoon, K. J. (2017). VPGNet: Vanishing point guided network for lane and road marking detection and recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 1951-1959).

7. Pan, X., Shi, J., Luo, P., Wang, X., & Tang, X. (2018). Spatial as Deep: Spatial CNN for Traffic Scene Understanding. In Proceedings of the European Conference on Computer Vision (ECCV) (pp. 768-784).

8. Li, Y., & Hoiem, D. (2018). Learning without human keypoints. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 41(9), 2251-2264.

9. Hou, Y., Ma, J., Liu, C., & Loy, C. C. (2019). Learning lightweight lane detection CNNs by self attention distillation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 6733-6742).

10. Qiu, Y., Wang, B., Shen, X., & Wang, Y. (2020). Real-time lane detection on the earth mover's distance transformed bird's-eye view. IEEE.