WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Road Lane Line Detection

***Patil Harshal Avinash*[1], *Sakre Divya Dilip*[2], *Kadam Aaditi Sarjerao*[3]**

[1,2,3]Department of Information Technology Vidyavardhini's College of Engineering and Technology Vasai, India

ABSTRACT—

Road lane line identification is regarded as a crucial component in the creation of autonomous vehicles that can recognise lanes and follow the road correctly without a driver's sight. We listed various kinds of investigations we conducted in this paper and selected the one that generated the most favorable lane detection results. We explored two techniques referred to as object detection and edge detection techniques. We learned from the results of these two procedures that the edge detection method will give us accurate results. The suggested method consists mostly of a pipeline of computer vision algorithms that build upon one another and consume raw RGB photos to generate the necessary lane-line segments that serve as the car's representation of the edge of the road. On actual videos, the performance of the entire pipeline is examined. The analysis of the suggested method demonstrates that it reliably detects and tracks road boundaries. The proposed method's benefits and drawbacks are thoroughly examined as well.

***Index Terms*—OpenCV, machine learning, image processing, neural network, detection, lane detection, autonomous cars, vehicles, safety, smart vehicle.**

I. INTRODUCTION

Road lane line detection, a key technique in the development of autonomous vehicles, makes use of computer vision, deep learning, and image processing. The technology of today's cars has advanced to include lane detection. This development is genuinely needed to tackle several problems encountered while driving which includes traffic congestion, autonomous driving, inattentive driving, and sudden road bends.

It may be a result of both the state of the roads or the drivers' disregard for traffic safety, but drivers who drive cars on the road occasionally have more serious accidents. Major traffic accidents can result from any careless or drowsy activity. Road lane detection might be a big assistance in managing these difficulties.

A significant number of cars move through busy cities every day in the same pattern. Most roads experience congested roads during rush hours because drivers don't abide by the rules and allow overtaking constantly. Yet, if each individual uses a road lane-detecting car, it will help to alleviate traffic congestion by maintaining the necessary spacing between the vehicles. Here, we see how important and challenging it is to integrate road lane detection.

II. METHODOLOGY

Road lane detecting is a procedure that may be learned and perfected. Deep image processing, deep learning, and machine learning work together to produce some expected outcomes. The first step of a road lane detection system is to identify any lanes that can be followed before moving the car in that direction. In order to detect the region of road, it is crucial that we first collect the video frames of a moving car. Further approach to apply detection methods over the region of interest that is road. We can split this approach into two components, one for object detection and the other for edge detection, to find the lanes.

A. Object Detection Methods

One method for finding instances of objects in pictures or videos is object detection. Item detection systems frequently use deep learning or machine learning to generate insightful results. In particular, bounding boxes are drawn around these items that are recognised via object detection, allowing us to determine their location in (or how they move across) a scene. Road lane detection can also be done using the object detection method. The fundamental method is to train the computer to recognise roads in photographs while ignoring ev- erything else. There are numerous ways for detecting objects, including the haar cascade, segmentation approaches, you only look once(yolo), neural network methods, etc. A very helpful reference for object detection is [3].

1. *Haar Cascade:* In spite of their size and location in the image, objects can be found in photographs using the Haar cascade technique. This algorithm can function in real time and isn't overly complicated. A haar-cascade detector can be trained to recognise many different items, including automo- biles, bikes, structures, fruits, etc. A very helpful reference for Haar Cascade is [7].

Here, a database of both positive and negative photographs was created. You can understand this by taking help of the figure number 1 and 2 which appears on page number 2. To program the machine and use it in the code, upload it to the Haar cascade trainer. Got Real-time computer vision using the opencv programming library. opencv is imported in the code, along with code of lines for drawing bounding boxes [7]. Significantly, contrary to our expectations, roads should only be present when bounding boxes exist. Yet, because of

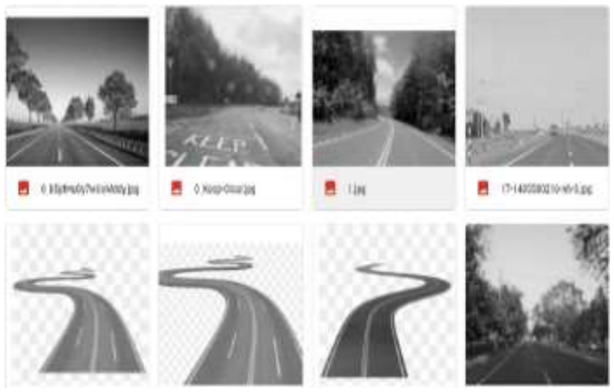

Fig. 1. Positive database containing road images.

Fig. 2. Negative database containing images except road images.

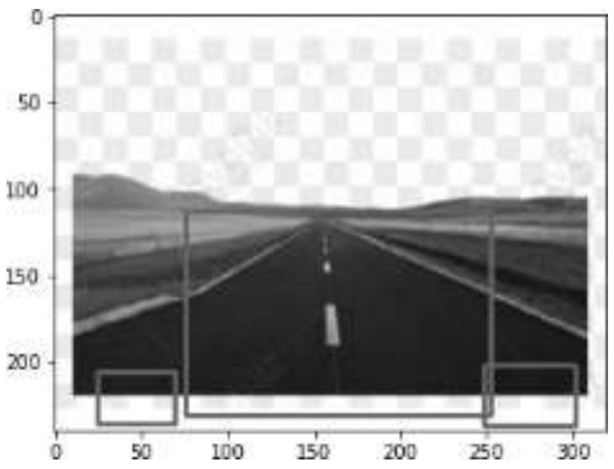

Fig. 3. Object detection output.

Surrounding items and various illumination conditions, it was challenging to identify roads as objects in the haar cascade.

2. *Image Segmentation Methods:* Image segmentation is a branch of computer vision and digital image processing that focuses on classifying related regions or segments of an image. Semantic, instance, and panoptic methods are the three segmentation techniques. We studied the semantic image segmentation techniques, which include categorising the pixels in an image into meaningful groups. Pixels that fall within a certain class are simply assigned to that class without taking any further context or data into account [4].

Here, the initial step was to gather pictures of the correct road lanes. the two databases Original Images and Binary

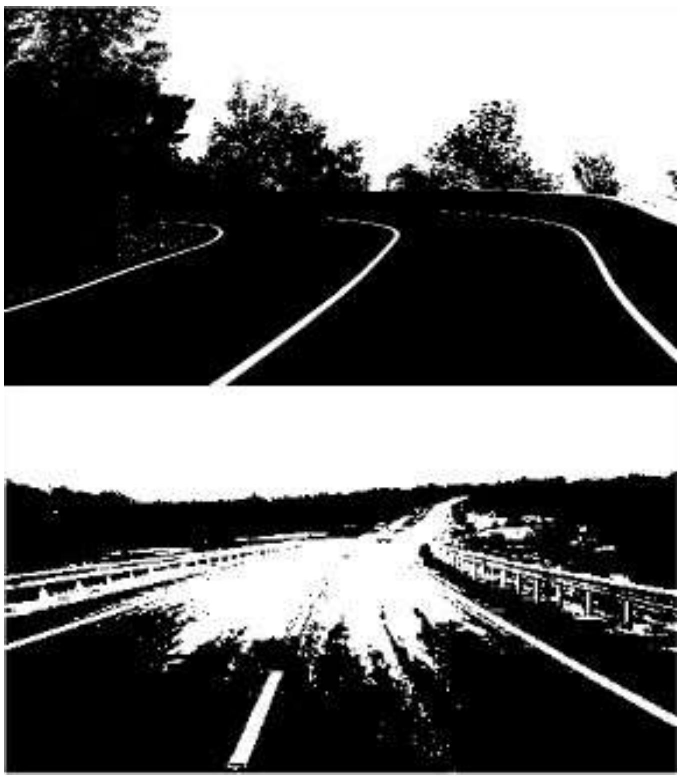
Fig. 4. Edge detection output

Images were principally produced. With the aid of Augmen- tation and Albumentation, they are further transformed into many images, which ultimately results in the images being sent through U-net. But in the phase that followed, we ran into a significant issue with the colour of the roads and the day and night lighting which lead to uncertain results.

Results of object detection can be seen in the figure number 3 and it appears on page number 2.

B. Edge Detection Method

The process of processing images that finds the edges of objects in images is known as edge detection. It works by scanning for variations in brightness. Edge detection is used for picture segmentation and data extraction in a variety of applications, such as image processing, computer vision, and machine vision.

In order to trace a polygon and aid in lane detection, the edge detection approach can be applied to identification of road lanes. It is attainable to look into edge detection techniques as canny edge, hough transform, ant colonisation, sliding window technique, RANSAC, etc. [1]

1. *Canny Edge with Hough Transform:* Firstly, with the use of the Canny edge detection technology, the amount of data that needs to be processed can be drastically reduced while still extracting meaningful structural information from various vision objects. It is frequently used in many computer vision systems. Secondly, if you can mathematically represent any shape, you can use the Hough Transform to detect it. Even if the shape is slightly fractured or warped, it can still be detected. We'll check how it functions for a line.

Straight lines may be found by using the clever edge detection technique to find polygons, then using the transform algorithm to trace the lane lines. Here, straight-road line detection has improved in accuracy.

We worked on both the object and edge detection methods to compare the outputs of each approach and choose the one that delivers the greatest accuracy. According to the findings, edge detection techniques produced excellent results.

Results of object detection can be seen in the figure number 4 and it appears on page number 2.

III. PROPOSED METHOD

After testing various approaches, we came to the conclusion that edge detection methods should be used as the final option. Refer to the pictures below for an illustration of the steps to take.

A. Gaussian Blur

When compared to the original image, the produced image will be smaller. The reason behind this is that the bottom and right edge pixels can't be totally removed. The resulting image will be smaller than the source image, m-1 right hand pixels and n-1 bottom pixels cannot be used because the kernel cannot fully map the bottom and right border pixels. Before attempting to find and detect any edges, the first stage in clever edge detection is to remove any noise from the source image. Unwanted detail and noise are blurred and eliminated using the Gaussian filter.

B. *Canny Edge Detection*

This part will cover the method of canny edge detection, which we'll utilise to create a program that can identify the edges in a picture. We therefore look for regions in an image where there is a sharp change in hue and intensity.

It's crucial to remember that a picture can be understood as a matrix (an array of pixels). Light levels at specific points in the image are contained in each pixel. The intensity of each pixel is represented by numerical values, ranging from 0 to 255. Black represents no intensity (0), and 255 represents the highest intensity (white).

We can recognise edges in our image by enhancing gradi- ents. The difference in intensity values between neighbouring pixels characterises an edge. There is a bright pixel in the gradient image for each sudden shift in brightness or intensity that you notice. We discover the edges by tracing these pixels. We will now use this understanding to look for edges in our image. This procedure consists of three steps.

1. *Gray-Scaling:* To make processing our image easier, we converted it to grayscale. A coloured image has more than three values, but a grayscale image only has a single pixel intensity value (0 or 1). By doing this, the grayscale image will operate in a single channel, making it simpler and quicker for us to process than three-channeled colour images. Instead of setting the new variable equal to the picture, it is crucial to make a copy of the image variable. By doing this, you can be sure that the modifications you make to lane image won't have an impact on image.
2. *Reduce Noise & Smoothen:* The image's edges should all be recognised as clearly as possible. Nevertheless, we also need to remove any image noise that can lead to spurious edges and ultimately impair edge identification. The filter known as Gaussian Blur will be used to reduce noise and smooth the image.

Remember that an image is stored as a collection of unique pixels. Each pixel in a gray-scale image is identified by a single number that describes its brightness. A pixel's value must be changed along with the average value of the pixels around it in order to smooth an image. A kernel will be used to average pixels in order to lower noise. This normally distributed kernel window is applied to our entire image, smoothing it by setting each pixel's value to the weighted average of its nearby pixels. We use the cv2.GaussianBlur() method on our grayscale image to encode this convolution.

Here, we're applying a kernel window of size 5 x 5 to our image. A 5*5 window is ideal in the majority of situations, while the kernel size varies on the circumstances. To obtain the output, we supply the blur variable to imshow(). Convo- lution of a kernel of noise-reduced Gaussian values yields a blurred image as a result.A very helpful reference for noise reduction [2].

3. *Canny Method:* Any additional lines found in step one that are outside of the area of interest should be omitted when you trace the Region Of Interest. The region of interest is shown in the figure number 5 and it appears on page number 4.You must keep in mind that a picture can also be represented in a 2D coordinate system using the letters X and Y in order to grasp this idea.

In an image, X represents the width (number of columns), and Y represents the height (number of rows). The sum of an image's width and height tells us how many pixels there are overall. This shows us that we can represent images as a continuous function of X and Y. We can undertake procedures to ascertain quick changes in the brightness of the pixels in an image since f(x, y) is a mathematical function. Our function's derivative in both the x and y axes will be provided via the canny approach. This derivative can be used to calculate the intensity change between two neighbouring pixels. Intensity and derivative changes are inversely correlated and vice versa. We determine the gradient of the image by computing the derivatives in all directions. All of these tasks are carried out for us by calling the cv2.Canny() method in our code. The strongest gradient will be represented as a series of white pixels using this function. We can isolate the nearby pixels that follow the strongest gradience using the two variables low threshold and high threshold. If the gradient is higher than the higher threshold, the pixel is categorised as an edge pixel. If it is below the lower level, it is rejected. It is only approved when the gradient between the thresholds is connected to a strong edge. In this instance, we'll use a low-to-high threshold ratio of 1:3. This time, instead of the blurred image, we get output as a clever image. A very helpful reference for Canny Method is [6]

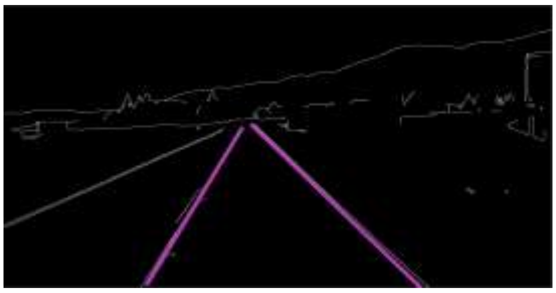

Fig. 5. Region of Interest

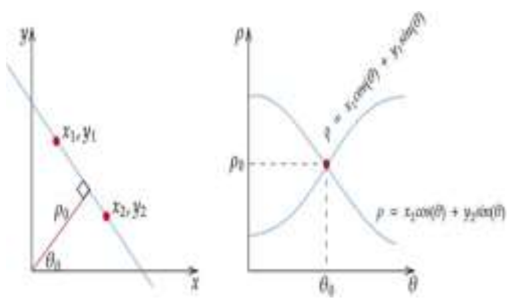
y
x_1, y_1
x_2, y_2
ρ_0
θ_0
x
ρ
$\rho = x_1\cos(\theta) + y_1\sin(\theta)$
ρ_0
$\rho = x_2\cos(\theta) + y_2\sin(\theta)$
θ_0
θ

Fig. 6. Sinusoidal Curve

C. Hough Transform Technique

As the image we currently have is just a set of pixels, it is difficult to discover a geometrical representation that would allow us to calculate the slope and intercepts.

Given that no photo is ever perfect, looping through the pixels to get the slope and intercept would be quite difficult. The hough transform can be applied in this situation. It allows us to recognise sharp edges and connect the image's disjointed edge points.

Let's compare the hough space to the conventional X-Y coordinate system to better understand this concept (M-C space).

An area in the X-Y plane can have any number of lines travel through it. Consider a line in the same X-Y plane such as several points cross it.

To identify the lines in our image, first consider each edge pixel as a point in a coordinate space and then convert that point into a line in the hough space.

To detect the lines, we must locate two or more lines that represent their corresponding points (in the X-Y plane) and intersect in the hough space. By doing so, we may determine that the two locations are part of the same line.

We can locate the lines in our gradient image using the concept of deriving potential lines from a set of points. Yet in order to detect the lines, the model also needs their parameters. To obtain these characteristics, the hough space is first divided into a grid of tiny squares, as seen in the picture. The best fit line will be drawn using the values of c and m that correspond to the square with the greatest number of intersections.

Sloped lines respond well to this strategy. Yet, since the slope of a straight line is always infinite, we are unable to precisely manipulate that value. So, we slightly alter our strategy.

Instead of representing our line equation as,

$y = mx + c$ (1)

We state our line equation in the polar coordinate system, i.e.,

$\rho = X\cos\theta + Y\sin\theta$ (2)

where:

ρ = perpendicular distance from the origin.

θ = angle of inclination of the normal line from the x- axis.

Instead of straight lines in the hough space, we obtain a sinusoidal curve when the line is represented in polar coordinates.The sinusoidal curve appears in figure number 6 and it appears on page number 4. All possible values for and of the lines that pass through our points are used to identify this curve. There are more curves in our hough space if we have more points. The numbers corresponding to the curves that overlap most frequently will again be used to determine the best fit line.A very helpful reference for Hough Transform [5].

D. Implementing Hough Transform

Let's use it in the code now that we have a method to recognise the lines in our image. Fortunately, we can utilise opencv's built-in method cv2.HoughLinesP() to accomplish this work.

The earlier-generated cropped image, which is an isolated gradient image of lane lines, serves as the first counterar- gument. The resolution of the hough accumulator array is specified by the second and third inputs (the grid created to recognise most intersections). The threshold value needed to determine the bare minimum of votes required to detect a line is the fourth input.

Let's construct a few functions to symbolise these lines before we show them in our actual image. To completely optimise and display the lane lines, we will define 3 functions.

- display lines: With this feature, lines will be drawn on a black image with dimensions that correspond to the source image before being blended into our colour image. To merge the line picture and the colour image, we use cv2.addWeight().
- To be able to identify the slope and y-intercept, we must first specify the coordinates.
- After that, we create two empty lists named left fit and right fit that will each hold the coordinates of the averagelines on the left and the right. This is known as the average slope intercept. The np.polyfit() method returns a vector of coefficients that contains information about the slope, y-intercept, and other characteristics of straight lines that would be a good fit for our points.

IV. RESULTS AND DISCUSSIONS These are the necessary software configurations:

Operating Systems: Mac OS, Linux, Windows 10/8/7

(including 64-bit), Language: Python 3

IDE: Spyder 3 with Jupyter Notebook Structure: Tkinter

These are the necessary minimal hardware setups.: Processor: Intel core 2 duo or newer processor required. Memory: 1 GB or more

HDD: at least 256 GB

Monitor: minimum 1024 × 768 resolution.

The many processes required to build our pipeline, which will allow us to recognise and categorise lane lines, will be covered in detail in this section. The pipeline will look like this: HSL conversion of the original image white and yellow from the HSL image. Grayscale an image for simpler manipulation Using Gaussian Blur, amplify edges Smoothed grey image with Canny Edge Detection applied Trace the Region Of Interest and omit any further lines discovered in step one that are not inside this region. Have a look Find the lanes in our area of interest using the Hough Transform and highlight them in red. Create two smooth lines by separating the left and right lanes. Results can be seen in figure number 7 and it appears on page number 5.

The evaluation of lane detection studies has revealed that the majority of researchers have disregarded the issue of the im- age's abstract lighting and road hue. As a result, illumination and road colour could make the current systems less accurate. Via the object detection method, one can further enhance the results by using colour balancing techniques.

V. CONCLUSION AND FUTURE WORK

Furthermore, the pipeline employs a thorough lane line recognition and drawing technique to create the result. The suggested method just requires raw RGB photos from a single CCD camera situated behind the car's front windscreen. With the help of numerous real-time movies and a large number of still photos, the Lane detection performance is tested and assessed.

We are now in the project's intermediate stage, and addi- tional features will be added soon.

ACKNOWLEDGMENT

We would like to thank the authors whose research we refer- enced and cited in this paper for their valuable contributions to the field of road lane line detection. Their work has provided us with a strong foundation to build upon and has added to the body of knowledge in this field.

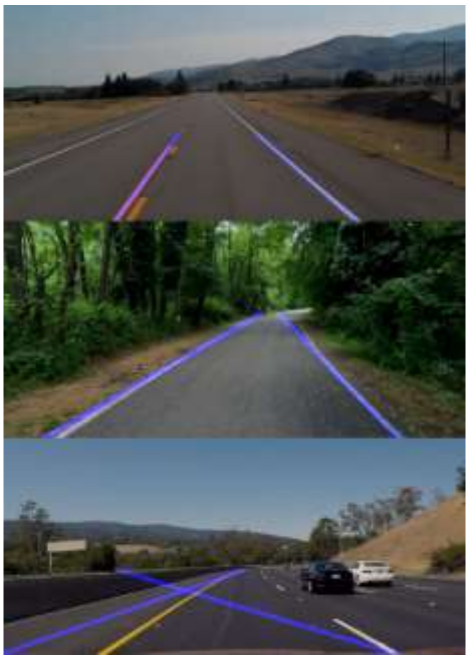

Fig. 7. Lane Detection performed on testing video one

REFERENCES

[1]. Ghassan Mahmoud Husien Amer and Ahmed Mohamed Abushaala. "Edge detection methods". In: *2015 2nd World Symposium on Web Applications and Networking (WSWAN)*. 2015, pp. 1–7. DOI: 10.1109/WSWAN.2015. 7210349.

[2]. Kanika Garg and Goonjan Jain. "A comparative study of noise reduction techniques for automatic speech recogni- tion systems". In: *2016 International Conference on Ad- vances in Computing, Communications and Informatics (ICACCI)*. 2016, pp. 2098–2103. DOI: 10.1109/ICACCI. 2016.7732361.

[3]. Manjula S. and Lakshmi Krishnamurthy. “A study on object detection”. In: *International Journal of Pharmacy and Technology* 8 (Dec. 2016), pp. 22875–22885.

[4]. Yuheng Song and Hao Yan. “Image Segmentation Tech- niques Overview”. In: *2017 Asia Modelling Symposium (AMS)*. 2017, pp. 103–107. DOI: 10.1109/AMS.2017.24.

[5]. Sumir Srivastava and Sumita Gupta. “Path Detection for Self-Driving Carts by using Canny Edge Detec- tion Algorithm”. In: *2021 9th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*. 2021, pp. 1–5. DOI: 10.1109/ICRITO51393.2021.9596109.

[6]. Zhao Xu, Xu Baojie, and Wu Guoxin. “Canny edge detection based on Open CV”. In: *2017 13th IEEE International Conference on Electronic Measurement In- struments (ICEMI)*. 2017, pp. 53–56. DOI: 10 . 1109 / ICEMI.2017.8265710.

[7]. Ratna Yustiawati et al. “Analyzing Of Different Features Using Haar Cascade Classifier”. In: *2018 International Conference on Electrical Engineering and Computer Science (ICECOS)*. 2018, pp. 129–134. DOI: 10 . 1109 / ICECOS.2018.8605266.