International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 09 Issue: 04 | Apr 2022 www.irjet.net p-ISSN: 2395-0072

LANE DETECTION USING IMAGE PROCESSING IN PYTHON

Nidhi Lakhani[1], Ritika Karande[2], Deep Manek[3], Vivek Ramakrishnan[4]

[1-3]BE Student, Electronics and Telecommunication, Atharva College of Engineering, Mumbai, India
[4]Professor, Electronics and Telecommunication, Atharva College of Engineering, Mumbai, India

---

Abstract - *When we drive, we use our eyes to decide where to go. The lines on the road portray us where the lanes act as our steady reference for where to steer the vehicle. Naturally, one of the first things we would like to develop a self-driving car is to detect lane lines using an algorithm automatically. Lane detection is a difficult problem to solve. For decades, it has piqued the interest of the computer vision community. Lane detection is essentially a multi-feature detection problem that has proven to be difficult for computer vision and machine learning algorithms to solve. We present an Image Processing based method that involves Region Masking and Canny Edge Detection. The prime goal is to use the canny edge detection algorithm to extract the features and then use another method to detect the lanes on the region masked image. As the images are not perfect every time, looping through the pixels in order to find the slope and intercept would be a difficult task. This is where we looked at the Hough transform concept. It aids us in identifying prominent lines and connecting disjoint edge points in an image.*

Key Words: Lane Detection, Image Processing, OpenCV, Hough Transform, Canny Edge Detection.

I. INTRODUCTION

Automobiles have become one of the transportation tools for people to travel due to the rapid development of society. There are an increasing number of vehicles of various types on the narrow road. Lane detection is a prominent topic in the field of machine learning and computer vision and it has been utilized in intelligent vehicle systems [1]. It is an emerging field and finds its applications in the commercial world. The lane detection system comes from the lane markers in an intricate environment and is used to approximate the vehicle's position and trajectory relative to the lane reliably [2]. At the same time, lane detection plays a substantial part in the lane departure warning system. The task of lane detection is mainly split into two parts: edge detection and line detection. Line detection is equally important as edge detection in the process of lane detection.

The most popular lane line detectors are the Hough transform and convolution-based techniques. Lane detection is the course of detecting lane markers on the road and thereafter presenting these locations to an intelligent system. Intelligent vehicles cooperate with the infrastructure to achieve a safer environment and better traffic conditions in intelligent transportation systems. The utilization of a lane detecting system could be as simple as pointing out lane locations to the driver on an external display to more intricate tasks such as predicting a lane change in an instantaneous moment to avoid collision with other vehicles.

II. GENERAL DESCRIPTION

In this project we have detected lane lines in images using Python as it is one of the widely used programming languages for this purpose and OpenCV. OpenCV stands for "Open-Source Computer Vision", which is a module that has numerous useful tools for analysis of images [3]. OpenCV supports a wide variety of programming languages like Python, and Java. It can assess images and videos to recognize objects, faces, or even the handwriting of a human. In this project, we also use NumPy for a few simpler image processing techniques.

The lane detection module is divided into two steps:

(1) Gaussian Smoothing and Canny Edge Detection and

(2) ROI selection and detecting lane lines using Hough Transform.

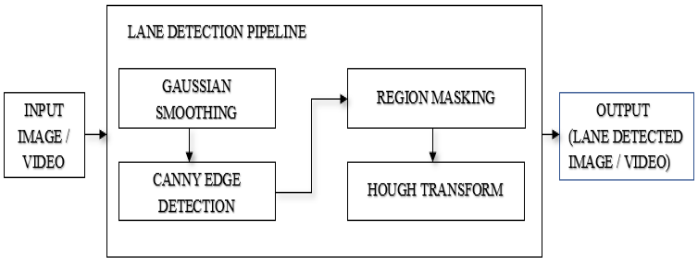

Figure-1: Block Diagram

II.A. GAUSSIAN SMOOTHING

Gaussian blur, in image processing, is also known as Gaussian smoothing. It is the outcome of blurring an image by a Gaussian function [4]. Noise reduction is gained by blurring the image with the support of a smoothing filter. Gaussian Smoothing is a widely used effect in graphics software, typically to reduce image noise. The 'kernel' for smoothing actually expresses the shape of the function that is used to compute the average of the neighboring points. A Gaussian kernel is a kernel which has the shape of a Gaussian i.e., a normal distribution curve. In order to use the Gaussian filter, we will first convert the image to grayscale so that it is easier for us to distinguish the output we get after running canny edge detector.

II.B. CANNY EDGE DETECTION

Edge detection is an image processing technique used to recognize points in a digital image with discontinuities, basically, sharp changes in the image brightness. These points where the image brightness varies sharply are called the edges (or boundaries) of the image. Canny edge [5] detector is probably the most commonly used and most effective method, because it uses first a gaussian filter, it has more noise immunity than other methods. There is a multitude of information available in the documentation of OpenCV itself. Thus, we decided to use Canny Edge Detection for this project.

To summarize, first, the algorithm will detect strong edge (strong gradient) pixels above the high threshold and reject pixels below the low threshold. Next, pixels with values between the low threshold and high threshold will be included as long as they are connected to strong edges. The output edges are a binary image with white pixels tracing out the detected edges and black everywhere else. We will also include Gaussian smoothing before running Canny, which is basically a way of suppressing noise and spurious gradients by averaging. cv2.Canny() function by OpenCV actually applies Gaussian smoothing internally, but we include it explicitly here because we can get a different result by using further smoothing and it's not a changeable parameter within cv2.Canny().

II.C. REGION MASKING

In an image, a region is a group of connected pixels that have similar properties. Because they may correlate to a items in a scene, regions are significant for image interpretation. Region Masking [6] is the process that is underneath many types of image processing, including edge detection, motion detection, and noise reduction. Masking a region is a view-specific graphic that can be used to hide certain and show certain elements in a view. It is a non-destructive process of editing images. Assuming that the camera (which is front-facing) that captures the image is mounted in a fixed spot on the car in a way that the lane lines will always appear in the same general section of the image [7]. Here, we use a triangular mask to illustrate the simplest case, but we can use a quadrilateral, and in principle, we could use any polygon. We would now use edge detection algorithm for detecting lane lines rather than relying on color. This would be a more reliable and efficient model.

II.D. HOUGH TRANSFORM

Now, to select lane lines in the edge detected image, we use the Hough Transform. The Hough Transform is a transform used to detect straight lines [8]. As per the documentation, to apply the transform, it is first desirable to have an edge detection pre-processing. In essence, a line can be spotted by finding the number of intersections between curves [9]. The more curves intersecting means that the line represented by that intersection have more points. In general, we can describe a threshold of the minimum number of intersections needed to detect a line. This is what the Hough Transform does. It keeps track of the intersection between curves of every point in the image. If the number of intersections is over a certain threshold, then it declares it as a line.

II.E. LANE DETECTION PIPELINE

The usual lane line detection method is to take an image of the road first with the aid of a camera fixed in the vehicle. Then the image taken is converted to a grayscale image to minimize the processing time. Secondly, the appearance of noise in the image will prevent the detection of the correct edge. Hence, the filters should be applied to remove noises like Gaussian filter. Then, the edge detector produces an edge image by using a canny filter with automatic thresholding to obtain the edges. Next, the edged image would be sent to the line detector after detecting the edges. Further, the Hough Transform is used to detect the required polygon on an edge detected image. Once the image goes through all these intermediate steps, a pipeline is made using these steps as functions with a single argument which is the source image. Once the testing and validation parts on images are over, we can move ahead with applying the same concept on lane videos. Here, each frame of the video is treated as an image and it goes through the lane detection pipeline. This step is repeated for all the frames in the videos and later all these frames are merged to create a single video file with the lane detected output.

Now that we have established the techniques and functions we need to use, we have created a lane detection pipeline. Through this function, our input image will go through all the required steps mentioned above.

II.F. IMPLEMENTATION

We will use Jupyter Notebook for code building purpose and to input the video and image file. Then, we will show the output file as a small GUI (Graphical User Interface) application written in Python. For this purpose, we use tkinter and moviepy library of Python.

III. RESULTS AND OUTPUT

Here, Figure 1 shows the output as a lane detected image on our test images and Figure 2 shows the output when our code was run on the test video.

Figure-2: Output images

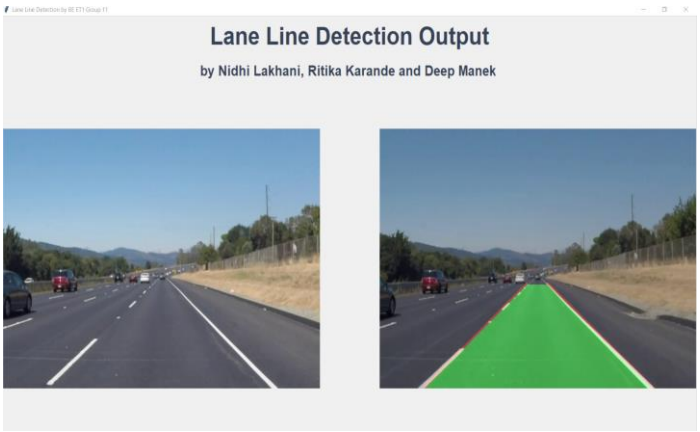

Figure-3: Video output screenshot

IV. APPLICATIONS

It can be used for lane detection in a driverless car that isn't fully automated so that it can assist the driver.

V. ADVANTAGES

- It assists vehicular drivers as well as autonomous cars to drive safely.
- This does not involve complex and advanced machine learning, deep learning or neural networks. Rather, relies on easy-to-understand and simple image processing and computer vision techniques.
- It has minimal cost of development as it uses open source technologies such as Python, Jupyter and OpenCV.

VI. DISADVANTAGES

- Uniformity of the markings present on the lanes is an important factor in order to minimize confusion and uncertainty. Our model needs lanes where there are uniform and visible markings.
- The methods that have been developed so far work efficiently and give good results in cases where noise is not present in the image. But the problem arises when lot of noise is in the road images, as it does not give efficient results.

VII. CONCLUSION

With all the developments in the autonomous vehicle industry, lane detection systems are going to be more in demand. Through this project, we effectively tried to propose the lane detection code without having to write long codes that use machine learning or deep learning. Through the Hough transform, the robustness and adaptability of the detection results are enhanced. There is some need for good dataset for rural road images which will broaden the application area of our project. The presence of noise is just one of the drawbacks but it can be rectified using good noise minimizing algorithms. We have not used object detection and tracking but it can be used to further incorporate more features into the algorithm.

VIII. FUTURE SCOPE

To further improve the robustness of the algorithm, some other methods can be considered in the future, such as, lane detection can be implemented using guided image filtering technique, for example, Gabor filter. Another issue to be dealt with is fog removal. In the near future, one can modify the existing Hough Transformation to measure both the curved and straight roads. Various steps should be taken to improve the results in different environmental conditions like sunny, foggy, rainy, etc. Further, the system can be expanded to include not only lane lines, but also road sign recognition as well as lane detection on rural or lesser travelled roads.

ACKNOWLEDGEMENT

We would like to give special thanks to our guide Prof. Vivek Ramakrishnan for mentoring us throughout our project. Also, we are thankful to the faculty of Electronics and Telecommunication Engineering Department at Atharva College of Engineering (University of Mumbai) for assisting and helping us with our work whenever needed.

REFERENCES

[1] Jianfeng Zhao, Bodong Liang, and Qiuxia Chen, "The key technology toward the self-driving car.", International Journal of Intelligent Unmanned Systems, Vol. 6, No. 1 (2018).

[2] Mohamed Aly, "Real Time Detection of Lane Markers in Urban Streets.", IEEE Intelligent Vehicles Symposium,

Eindhoven University of Technology, Eindhoven, The Netherlands (2008).

[3] https://docs.opencv.org/4.x/

[4] Hongwei Guo, "A Simple Algorithm for fitting a gaussian Function," IEEE Signal Processing Magazine, September 2011.

[5] Y. Wang, E. K. Teoha, and D. Shen., "Lane detection and tracking using B-Snake.", Image and Vision Computing 22, pp: 269-28 (2004).

[6] https://www.cse.unr.edu/~bebis/CS791E/Notes/AreaProcess.pdf

[7] Raman Shukla et al., "Lane Detection System in Python using OpenCV," IJIRT, Volume 8 Issue 1, June 2021.

[8] F. Mariut, C. Fosalau and D. Petrisor, "Lane Mark Detection Using Hough Transform.", IEEE International Conference and Exposition on Electrical and Power Engineering, pp. 871 – 875 (2012).

[9] K. Ghazali, R. Xiao, and J. Ma, "Road Lane Detection Using H-Maxima and Improved Hough Transform.", Fourth International Conference on Computational Intelligence, Modelling and Simulation, pp. 2166-8531 (2012).