Lane line Detection System in Python using OpenCV

Raman Shukla[1], Rajat Shukla[2], Sarthak Garg[3], Sharad Singh[4], Pooja Vajpayee[5]
[1,2,3,4,5]*Computer Science, Raj Kumar Goel Institute of Technology, Uttar Pradesh, India*

Abstract **- During the driving operation, humans use their optical vision for vehicle maneuvering. The road lane marking, act as a constant reference for vehicle navigation. One of the prerequisites to have in a self-driving car is the development of an Automatic Lane Detection system using an algorithm.**

Computer vision is a technology that can enable cars to make sense of their surroundings. It is a branch of artificial intelligence that enables software to understand the content of image and video. Modern computer vision has come a long way due to the advances in deep learning, which enables it to recognize different objects in images by examining and comparing millions of examples and cleaning the visual patterns that define each object. While especially efficient for classification tasks, deep learning suffers from serious limitations and can fail in unpredictable ways.

This means that a driverless car might crash into a truck in broad daylight, or worse, accidentally hit a pedestrian. The current computer vision technology used in autonomous vehicles is also vulnerable to adversarial attacks, by manipulating the AI's input channels to force it to make mistakes. For instance, researchers have shown they can trick a self-driving car to avoid recognizing stop signs by sticking black and white labels on them.

Index Terms **- Deep learning (DL), Machine Learning (ML), Convolutional neural networks, Computer Vision.**

I.INTRODUCTION

Traffic accidents have become one of the most serious problems in today's world. Roads are the mostly chosen modes of transportation and provide the finest connections among all modes. Most frequently occurring traffic problem is the negligence of the drivers and it has become more and more serious with the increase of vehicles. These road accidents can be reduced with the help of road lanes or white markers that assist the driver to identify the road area and nonroad area. A lane is a part of the road marked which can be used by a single line of vehicles as to control and guide drivers so that the traffic conflicts can be reduced.

Increasing the safety and saving lives of human beings is one of the basic function of Intelligent Transportation System (ITS). Intelligent transportation systems are advanced applications which aim to provide innovative services relating to different modes of transport and traffic management. This system enables various users to be better informed and make safer, more coordinated, and smarter use of transport networks. These road accidents can be reduced with the help of road lanes or white markers that assist the driver to identify the road area and non-road area. A lane is a part of the road marked which can be used by a single line of vehicles as to control and guide drivers so that the traffic conflicts can be reduced.

Most roads such as highways have at least two lanes, one for traffic in each direction, separated by lane markings. Major highways often have two roadways separated by a median, each with multiple lanes. To detect these road lanes some system must be employed that can help the driver to drive safely.

Lane Line detection is a critical component for self driving cars and also for computer vision in general. This concept is used to describe the path for self-driving cars and to avoid the risk of getting in another lane. Using computer vision techniques in Python, we will identify road lane lines in which autonomous cars must run. This will be a critical part of autonomous cars, as the self-driving cars should not cross it's lane and should not go in opposite lane to avoid accidents.

II. LITERATURE REVIEW

Despite the perceived simplicity of finding white markings on a simple road, it can be very difficult to determine lane markings on various types of road. These difficulties can be shadows, occlusion by other vehicles, changes in the road surfaces itself, and different types of lane markings. A lane detection

system must be able to detect all manner of markings from roadways and filter them to produce a reliable estimate of the vehicle position relative to the lane. To detect road markings and road boundaries various methodologies are used like Hough Transform, Canny edge detection algorithm, bilateral filter. The main working of all these are as follows:

2.2.1 Hough Transform

The Hough transform is a technique in which features are extracted that is used in image analysis and digital image processing. Previously the classical Hough Transform worked on the identification of lines in the image but later it has been extended to identifying positions of shapes like circles and ellipses. In automated analysis of digital images, there was a problem of detecting simple geometric shapes such as straight lines, circle, etc[1]. So in the pre-processing stage edge detector has been used to obtain points on the image that lie on the desired curve in image space. But due to some imperfections in image data or in the edge detector, some pixels were missing on the desired curve as well as spacial deviation between the geometric shape used and the noisy edge pixels obtained by the edge detector. So to refine this problem Hough transform is used. In this the grouping of edge pixels into an object class is performed by choosing appropriate pixels from the set of parametric image objects[2]. The simplest case of Hough transform is finding straight lines that are hidden in large amounts of image data. For detecting lines in images, the image is first converted into binary image using some form of thresholding and then the positive or suitable instances are added into the dataset. The main part of Hough transform is the Hough space. Each point (d, T) in Hough space matches to a line at angle T and distance d from the origin in the data space. The value of a function in Hough space gives the point density along a line in the data space.

2.2.2 Edge Detection

Edge detection works on the idea of the identification of points in the digital image at which the image brightness changes sharply. The points at which image brightness changes sharply are organized into a set of curved line segments termed as edges. Edge detection is a fundamental tool in image processing particularly in the areas of feature detection and extraction. Applying an edge detection algorithm to an image may significantly reduce the amount of data to be processed and may therefore filter out information that may be regarded as less relevant, while preserving the important structural properties of an image. If the edge detection step is successful, the subsequent task of interpreting the information contents in the original image may therefore be substantially simplified. However, it is not always possible to obtain such ideal edges from real life images of moderate complexity. The Canny edge detector is an edge detection algorithm that uses a multiple stage algorithm so as to detect edges in images[3]. Its aim is to discover the optimal edge detection. In this definition, an optimal edge detector includes the following things

- Good detection – the algorithm should be able to detect as many real edges in the image as possible.
- Good localization – edges marked through this algorithm should approach as close as possible to the edge in the real image.
- Minimal response – a given edge in the image should only be marked once so as to reduce false edges

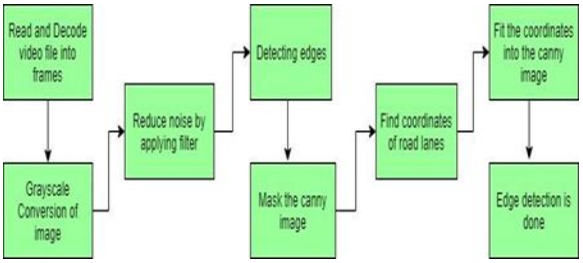
2.2.3 Bilateral filter

Bilateral filter is a simple and non-iterative scheme which smoothens the image while preserving the edges. The basic idea behind the working of bilateral filter is that the two pixels should be close to one another[4]. This filter split an image into large-scale features i.e. structure and small scale features i.e. texture. In this filter every sample is replaced by a weighted average of its neighbors. These weights reflects two forces i.e. the closeness of the neighborhood with the center sample so that larger weight is assigned to the closer samples, and similarity between neighborhood and the center sample so that larger weight is assigned to the similar samples[5].

III. METHODOLOGY

When we drive, we use our eyes to decide where to go. The lines on the road that show us where the lanes are, act as our constant reference for where to steer the vehicle. Naturally, one of the first things we would like to do in developing a self-driving car is to automatically detect lane lines using an algorithm[6]. In this project you will detect lane lines in images using Python and OpenCV. OpenCV means "Open-

Source Computer Vision", which is a package that has many useful tools for analyzing images.The tools we have are color selection, region of interest selection, grayscaling, Gaussian smoothing, Canny Edge Detection and Hough Tranform line detection. The stages of the proposed system are shown in Fig. Below



1. Color Selection

First let us select some colors. For Instance: Lane Lines are usually White in color and we know the RGB value of White is (255,255,255). Here we will define a color threshold in the variables red_threshold, green_threshold and blue_threshold and populate rgb_threshold with these values. This vector contains minimum values for red,green and blue(R,G,B).

2. Region Masking

Assuming that the front facing camera that took the image is mounted in a fixed position on the car, such that the lane lines will always appear in the same general region of the image. Next, taking advantage of this by adding a criterion to only consider pixels for color selection in the region where we expect to find the lane lines.

3. Canny Edge Detection

Now we are applying Canny to the gray-scaled image and our output will be another image called edges. low_threshold and high_threshold are your thresholds for edge detection.

4. Hough Transform

Now that we have detected edges in the region of interest, we want to identify lines which indicate lane lines. This is where the hough transform comes in handy.The Hough transformation converts a “x vs. y” line to a point in “gradient vs. intercept” space. Points in the image will correspond to lines in hough space. An intersection of lines in hough space will thus correspond to a line in Cartesian space. Using this technique, we can find lines from the pixel outputs of the canny edge detection output.

IV. CONCLUSION

In this paper, we proposed the approach to detect lanes, detect and track multiple vehicles for lane change support around the test vehicle. For lane detection, to detect lane in real-time, we use EDLines algorithm which can detect line segments between 10 ms and 20 ms on 2.2 GHz CPU, and EDlines was applied to ROI.Therefore, lane detection method has been implemented in the3.3 GHz Intel CPU and it takes about 13 ms with each the image.

With frontal view, our algorithm detects three lane areas, frontallane, left-side lane. Moreover, both rear-side lane were detectedwith two rear-side cameras by using our method.

Based on the detected lane areas combined with horizontal edge feature of vehicles, vehicle candidates are detected in each lane, and then the vertical edge are used to verify the vehicle candidates. With wrong detection cases, we use Kalman filter to predict and track vehicle target. Time of vehicle detection is about 30 ms, total time for lane detection and vehicle detection are about 43 ms. From the outcome of experiments our method obtains the goal in support lane change and warning.

REFERENCES

[1] J. Long, E. Shelhamer, and T. Darrell, “LANE DETECTION TECHNIQUES” – A Review.” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 3431–3440.

[2] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, And P.H Torr, “A Layered Approach to Robust Lane Detection at Night.” 2015, pp. 1529–1537.

[3] V. Badrinarayanan, A. Handa, and R. Cipolla, “Segnet: A deep convolutional encoder-decoder architecture for robust semantic pixel-wiselabelling,” arXiv preprint arXiv:1505.07293, 2015.

[4] M.Cordts,M.Omran,S.Ramos,T.Rehfeld,M.Enzweiler,R.Benenson,U. Franke,S.Roth,and B.Schiele, “The cityscapes dataset for semantic urban scene understanding,” in Proc. of the IEEE Conference

on Computer Vision and Pattern Recognition (CVPR), 2016.

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," arXiv preprint arXiv:1512.03385, 2015.

[6] Shopa, P., N. Sumetha and P.S.K Pathra. "Traffic sign detection and recognition using OpenCV", International Conference on Information Communication and Embedded Systems (ICICES2014), 2014.