IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

ROBUST LANE DETECTION AND TRACKING ALGORITHM FOR STEERING ASSIST SYSTEMS USING COMPUTER VISION

M.Rizvana M.E.,1, Swetha.G 2,Tasmiya Anjum.N3,Thenmozhi.M4

#1 Assist.Professor, Dept.of IT , PSV College of Engineering and Technology,Krishanagiri,Anna University, Tamilnadu,India

#2 UGScholar,Dept.of IT,PSV College of Engineering and Technology,Krishanagiri,Anna University, Tamilnadu State,India

#3 UGScholar,Dept.of IT,PSV College of Engineering and Technology,Krishanagiri,Anna University, Tamilnadu State,India

#4 UGScholar,Dept.of IT,PSV College of Engineering and Technology,Krishanagiri,Anna University, Tamilnadu State,India

Abstract: Modern vehicles rely on a multitude of sensors and cameras to both understand the environment around them and assist the driver in different situations. Lane detection is an overall process as it can be used in safety systems such as the lane departure warning system (LDWS). Lane detection may be used in steering assist systems, especially useful at night in the absence of light sources. Although developing such a system can be done simply by using global positioning system (GPS) maps, it is dependent on an internet connection or GPS signal, elements that may be absent in some locations. Because of this, such systems should also rely on computer vision algorithms. In this paper, we improve upon an existing lane detection method, by changing two distinct features, which in turn leads to better optimization and false lane marker rejection. We propose using a probabilistic Hough transform, instead of a regular one, as well as using a parallelogram region of interest (ROI), instead of a trapezoidal one. By using these two methods we obtain an increase in overall runtime of approximately 30%, as well as an increase in accuracy of up to 3%, compared to the original method.

Keywords: computer vision, steering assist systems, lane detection, autonomous driving, Hough transform, Patterns thereon, automation of driving system, vanishing point detection, exploitation, Hough Transformation Space, Region of Interest, Canny Edge Detection.

1. Introduction

Lane detection and tracking are difficult tasks in computer vision because creating a stable and computationally efficient method is a difficult challenge. Figures 1 and 2 depict some of the reasons that can lead to erroneous outcomes. Methods for detecting and tracking lanes must also be optimal. Otherwise, they may be unsuitable for actual application.

Figure 1. Parapets that may be detected as actual lane markers

This section will show various lane detection and tracking algorithms, while the following section will examine one of the listed techniques in depth. Then, we'll look at its computational complexity and drawbacks. Finally, optimisation suggestions will be presented, and the outcomes will be compared to the basic technique.

As urban traffic expands, road safety becomes increasingly important. The bulk of accidents on the avenues are caused by people exiting lanes without following the rules. The bulk of these are the result of the driver's erratic and sluggish driving. Lane discipline is vital on the road for both automobiles and pedestrians. Computer vision is a type of technology that allows autos to understand their surroundings. It is a branch of artificial intelligence that assists software in understanding image and video input. The system's objective is to locate the lane markings. Its purpose is to make the environment safer and transportation conditions better.

Accurate detection of lane roads is critical in lane recognition and departure warning systems. When a vehicle crosses a lane boundary, vehicles equipped with the predicting lane borders system warn the driver and direct the vehicle to avoid a collision. Although these intelligent systems always provide safe travel, it is not always necessary for lane boundaries to be clearly visible, as poor road conditions, insufficient lane boundary paint, and other factors can make it difficult for the system to detect the lanes accurately. Environmental impacts such as shadows projected by objects such as trees or other autos, or street lights, day and night time circumstances, or fog created by unchanging lighting conditions can also be considered.

To solve the difficulties stated above as a result of lane border changes. The algorithm utilised in this work seeks to recognise lane lines on the road by giving the system a video of the road as an input, with the primary goal of reducing the frequency of accidents. Accidents on the road caused by irresponsible driving can be averted by installing a system in autos and taxis. It will safeguard the safety of students on school buses. Furthermore, the performance of the driver can be tracked, and Road Transport Offices can use the system to monitor and report driver irresponsibility and lack of attention on the roads.

2. Background study

[1] A. Sharma and M. Kumar propose a basic approach that is implemented in Python using the Open Computer Vision (OpenCV) package. To begin, the approach transforms the input photos to grayscale and then uses the Canny operator to locate edges. Following that, a trapezoidal region of interest (ROI) is employed to eliminate sky and side road objects, but it does not erase the space between the lane markers, which is likewise uninteresting to us and hence a disadvantage.The Hough transform is then employed to detect lines, although doing so for each edge pixel in every frame to discover lane markers is redundant and may result in inaccurate detection. The RANSAC modelling technique is used to remove noise from datasets. In this study, we will investigate this approach in depth and offer enhancements such as employing a parallelogram ROI and a probabilistic Hough transform.

[2] J. Cao et al. offer a lane recognition system for intelligent vehicles operating in difficult road situations. They convert the deformed image and employ the superposition threshold approach for edge detection in order to obtain an aerial view of the lane via region of interest extraction and inverse perspective translation. They fit the curves of lane lines based on the third-order B-spline curve model with the random sample consensus algorithm, evaluate the fit, and calculate the curvature radius.

[3] Q. Li et al. present a robust and efficient strategy for extending the use of vision-based lane-detection technologies to low-speed conditions. They set the reliable region near the car to zero and dynamically build a succession of rectangle detection regions along the route. They use improved symmetrical local threshold edge extraction to extract the lane markings' edge points based on accurate marking width constraints. They propose a new Bresenham line voting space to improve the process of detecting line segments. The proposed geometric fitting method can adapt to varied road shapes when combined with straight lines, polylines, and curves. They track the important points in the linear and nonlinear parts of the lane using various status vectors and Kalman filter transfer matrices.

[4] X. Zhang et al. offer an adaptive lane feature learning system that can learn lane features automatically in a variety of settings. They build a two-stage learning network based on YOLO v3 (you only look once, version 3) with the YOLO v3 algorithm's structural parameters adjusted to make it more suitable for lane detection.They offer a method for automatically generating lane label images in a simple situation, which provides label data for the first-stage network's training. To relocate the lane indicated by the first-stage model, they employ an adaptive edge detection approach based on the Canny operator. They employ the above-mentioned photos as label data for training the second-stage model.

3. Proposed Methodology

Materials and Methods

3.1. Lane Detection Algorithm

We will examine the method suggested in this part because of its simplicity and ease of implementation using the OpenCV library. The following paragraphs will go over the most essential approaches described in the paper. Furthermore, the information will be used to develop a rudimentary steering assist system. The next parts will discuss the results and improvements.

3.1.1. Image Filtering

To recognize lines accurately, the image must be preprocessed with a noise-removing filter. One option for our purposes is to blur the image with a Gaussian filter. The blurring of the image is accomplished through the convolution of the original image with a kernel with Gaussian values. The following equation represents the 2D Gaussian function:

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

The x and y inputs in the preceding equation represent distances from the origin on the horizontal and vertical axes, respectively, while denotes the standard deviation of the Gaussian distribution. Figure 3 depicts a graphical representation of the Gaussian function. Figure 4 depicts blurring using a 3x3 Gaussian kernel by computing for the scalar product of the kernel and a sliding window the same size as the kernel and cantered on the pixel for each pixel (the matrix is padded by border mirroring if the sliding window exceeds the image boundaries).

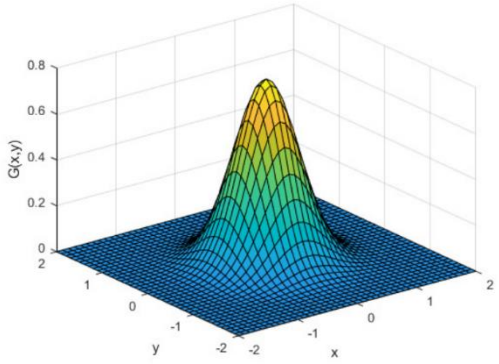

Figure 2. Gaussian function plot.

272	384	16	128	240
368	80	112	224	256
64	96	208	320	352
160	192	304	336	48
176	288	400	32	144

*

1/16	1/8	1/16
1/8	1/4	1/8
1/16	1/8	1/16

=

167	172	101	115	122
174	175	152	209	194
111	151	208	265	201
118	207	264	241	138
112	197	211	140	67

Figure 3. Gaussian convolution example

3.1.2 Canny Edge Detector

Canny edge detector [13] is a John F. Canny algorithm that can considerably reduce the number of erroneous edges detected in an image.

Four filters are employed to detect vertical, horizontal, and diagonal edges after applying a Gaussian filter, as explained in the preceding chapter. An edge detection operator is used to acquire the values for the first derivative in the vertical and horizontal directions. We can compute the edge gradient (2) and direction (3) using the derivatives. Finally, four angles are employed to express the horizontal, vertical, and both diagonal directions: 0, 90, 45, and 135 degrees.

$$G = \sqrt{G_x^2 + G_y^2}$$

$$\theta = atan2(G_x + G_y)$$

Edge thinning is the algorithm's next process. In the gradient image, we compare the edge strength of each pixel to the edge strength of its neighbours in both the negative and positive gradient directions. The value will be kept only if its edge strength is greater than that of the neighbouring pixel.

For example, if there is a vertical edge like the one in Figure 5, the red-framed pixel will be compared with the two black-framed pixels on the left and right.

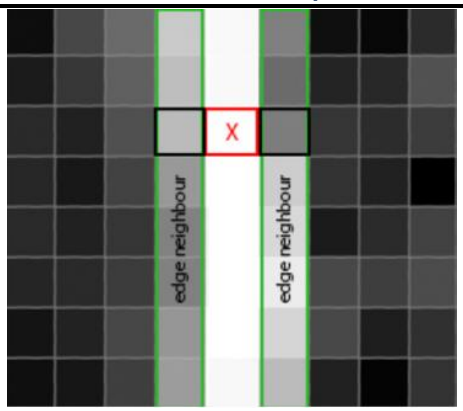

Figure 4. Example of a vertical edge

The Proposed Algorithm

To sum up the ideas presented in the last paragraphs we propose Algorithm 1:

Algorithm 1. The proposed lane detection algorithm.

Initialize parallelogram region of interest as undefined
For each recorded frame
Apply the Canny edge detector.
If parallelogram region of interest is undefined:
Define a trapezoidal region of interest for the road in front of the car based on the camera position.
Detect lines using the probabilistic Hough transform.
Detect two endpoints for each lane using the rectangle criteria.
If both lanes were detected:
Create a parallelogram region of interest for the two detected lines.
Calculate the slope for each line.
If slope is < 0:
Classify line as left_lane
Else
Classify line as right_lane
If left_lane slope > threshold_1
Indicate “Right turn”
If right_lane slope < threshold_2
Indicate “Left turn”
If !(left_lane slope > threshold_1) and !(right_lane slope < threshold_2)
Indicate “Go forward”
If left or right lane has not been detected
Define a trapezoidal region of interest for the road in front of the car based on the camera position.
Detect lines using the probabilistic Hough transform.
Detect two endpoints for each lane using the rectangle criteria.
If both lanes were detected:
Create a parallelogram region of interest for the two detected lines.
Calculate the slope for each line.
If slope is < 0:
Classify line as left_lane
Else
Classify line as right_lane
If left_lane slope > threshold_1
Indicate “Right turn”
If right_lane slope < threshold_2
Indicate “Left turn”
If !(left_lane slope > threshold_1) and !(right_lane slope < threshold_2)
Indicate “Go forward”
Else
Message “At least one lane has not been detected.”
Set parallelogram region of interest as undefined.

4. Implementation

Description General

In this project, we detected lane lines in photographs using Python, one of the most used programming languages for this purpose, and OpenCV. OpenCV stands for "Open-Source Computer Vision," and it is a package that contains a variety of useful tools for image analysis [3]. OpenCV is compatible with a wide range of programming languages, including Python and Java. It can analyse photos and movies to recognise items, faces, and even human handwriting. NumPy is also used in this project for a few simpler image processing techniques. The lane detection module consists of two steps: (1) Gaussian Smoothing and Canny Edge Detection, and (2) ROI selection and lane line detection using the Hough Transform. The lane detection module consists of two steps: (1) Gaussian Smoothing and Canny Edge Detection, and (2) ROI selection and lane line detection using the Hough Transform.

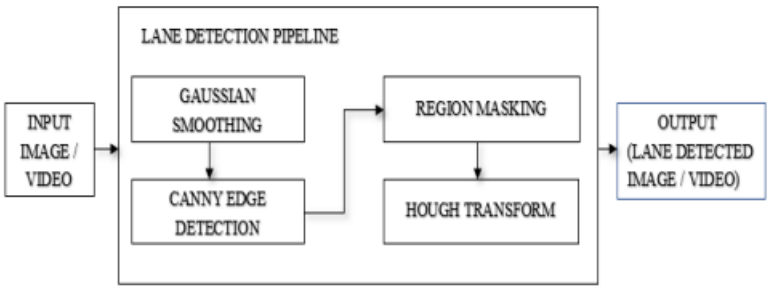

Fig.5.Architecture diagram

Gaussian Smoothing

Gaussian blur is also known as Gaussian smoothing in image processing. It's the result of blurring an image with a Gaussian function .Noise is reduced by blurring the image with the use of a smoothing filter.

Gaussian Smoothing is a popular effect in graphics software for reducing visual noise. The smoothing 'kernel' really expresses the shape of the function used to compute the average of the neighboring points. A Gaussian kernel is one that has the shape of a Gaussian, that is, a normal distribution curve. To utilize the Gaussian filter, we will first convert the image to grayscale so that we can identify the result after running it.

Canny Edge Detection

Edge detection is an image processing approach that identifies points in a digital image that have discontinuities, or sudden changes in image brightness. The image's edges (or boundaries) are the points where the image brightness fluctuates drastically. The Canny edge detector is arguably the most widely used and effective method since it uses a gaussian filter first and has better noise immunity than other methods. The documentation for OpenCV itself contains a wealth of knowledge. As a result, we chose Canny Edge Detection for this project.

To summaries, the algorithm will find pixels with strong edges (strong gradients) above the high threshold and discard pixels below the low threshold. Following that, pixels with values between the low and high thresholds will be included if they are connected to strong edges. The edges are a binary picture with white pixels tracing out the detected edges and black pixels everywhere else. Before running Canny, we will additionally apply Gaussian smoothing, which is a method of suppressing noise and false gradients through averaging. The OpenCV cv2.Canny() function uses Gaussian smoothing internally, but we include it directly here because we can get a different result by applying additional smoothing, and it is not a changeable parameter within cv2.Canny().

Region Masked

A region in a picture is a set of connected pixels with comparable attributes. Regions are important for picture interpretation because they may correspond to things in a scene.

Region Masking is a procedure that is used in many different types of image processing, such as edge detection, motion detection, and noise reduction. A area mask is a view-specific graphic that can be used to hide and reveal objects in a view. It is a non-destructive image altering method. Assuming that the front-facing camera that records the image is installed in a fixed location on the automobile so that the lane lines always display in the same general part of the image.

Hough Transfer

The Hough Transform is now used to pick lane lines in the edge detected image. The Hough Transform is a transform that is used to find straight lines. According to the documentation, edge detection pre-processing is required before applying the transform. A line can be found by counting the number of intersections between curves. The more curves that connect, the more points there are on the line represented by that intersection. In general, we can define a threshold as the number of intersections required to detect a line. This is what the Hough Transform accomplishes. It maintains note of where the curves of each point in the image cross. It indicates if the number of intersections exceeds a specified threshold.

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.

5. RESULTS AND OUTPUT

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.

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-6: Output images

Figure-7: Video output screenshot

6. 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.

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.

7. Acknowledgements

I first offer my thanks to all those who are graciously given me the strength and good health during the course of the project at thus pleasing moment of having successfully completed my project, I wish to convey our sincere thanks and gratitude to our beloved chairman, **Dr.P.SELVAM,M.A.,B.Ed.,M.Phil.,Ph.D.,** P.S.V. Group of Institutions.We also acknowledge our deep sense of gratitude to our secretary **Dr.S.VIVEK,M.A.,M.B.A(U.K).,Ph.D.,** P.S.V. Group of Institutions.I would like to express by sincere thanks to my beloved principal **Dr.P.LAWRENCE.,M.E.,Ph.D.,** P.S.V. College of Engineering and Technology, Krishnagiri, for forwarding us to our project and offering adequate duration in completing my project.I also express my sincere thanks to, Head of the Department, **Dr.M.SRINIVASAN,M.E.,Ph.d.,** Department of Information Technology, for providing all the facilities in the successful completion of my project. I have great pleasure to express my sense of gratitude to my internal guide **Prof.M.Rizvana,M.E.,** Assistant professor and Department of Information Technology, whose guidance and encouragement made this project an interesting educational experience.

.

REFERENCES

1.Sharma, A.; Kumar, M.; Gupta, R.K.; Kumar, R. Lane detection using Python. IJIRMPS 2021, 9, 917.

2. Cao, J.; Song, C.; Song, S.; Xiao, F.; Peng, S. Lane Detection Algorithm for Intelligent Vehicles in Complex Road Conditions and Dynamic Environments. Sensors 2019, 19, 3166. [CrossRef] [PubMed]

3. Li, Q.; Zhou, J.; Li, B.; Guo, Y.; Xiao, J. Robust Lane-Detection Method for Low-Speed Environments. Sensors 2018, 18, 4274. [CrossRef] [PubMed]

4. Zhang, X.; Yang, W.; Tang, X.; Liu, J. A Fast Learning Method for Accurate and Robust Lane Detection Using Two-Stage Feature Extraction with YOLO v3. Sensors 2018, 18, 4308. [CrossRef] [PubMed]