Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X
IMPACT FACTOR: 7.056

IJCSMC, Vol. 10, Issue. 4, April 2021, pg.01 – 05

LANE DETECTION USING COMPUTER VISION FOR SELF-DRIVING CARS

S.E.Viswapriya; Nagireddy Chaitanya; Kongara Sasi Harini
Assistant Professor, Student, Student
Department of Computer Science, SCSVMV University, Kanchipuram, India
DOI: 10.47760/ijcsmc.2021.v10i04.001

Abstract – In order to drive by themselves, they need to understand their surrounding world like human drivers, so they can navigate their way in streets and avoid hitting obstacles such as other cars and pedestrians. An effort has been made to demonstrate lane detection using Open-CV library based on the problems encountered in detecting objects by autonomous vehicles.
Keywords: NumPy, OpenCV, Canny, Lane-Detection, Hough Transform

I. INTRODUCTION

Humans use their optical vision for vehicle maneuvering during the driving operation. A constant reference for vehicle navigation is the road lane marking. The development of an Automatic Lane Detection system using an algorithm is one of the prerequisites to have in a self-driving car. A critical component of self-driving car is Lane detection.

II. LITERATURE SURVEY

We have studied several papers based on the lane detection techniques First paper which we have studied is named as Lane Detection for Autonomous Vehicles using OpenCV Library published by Aditya Singh Rathore, B.Tech, J.K.Lakshmipat University. This paper is about the techniques like canny edge detection, region of interest, Hough transform which we will be using to detect lane. The edges of object or lane is detected perfectly has been known. We can crop the region and also we can know how it is taken by using certain techniques. By using particular technique such as Hough Transform Technique we come to know about how the lane lines are detected. The third paper which we came across was titled as Lane intelligence that enables software to understand the content Detection on Roads using Computer.

III. EXISTING SYSTEM

In the existing Car system, We cannot prevent the car system from drifting off the driving lane. we are facing many problems like Heavy accidents, Safety issues etc, in existing system. The detection of location is flexible if an automated car is built.

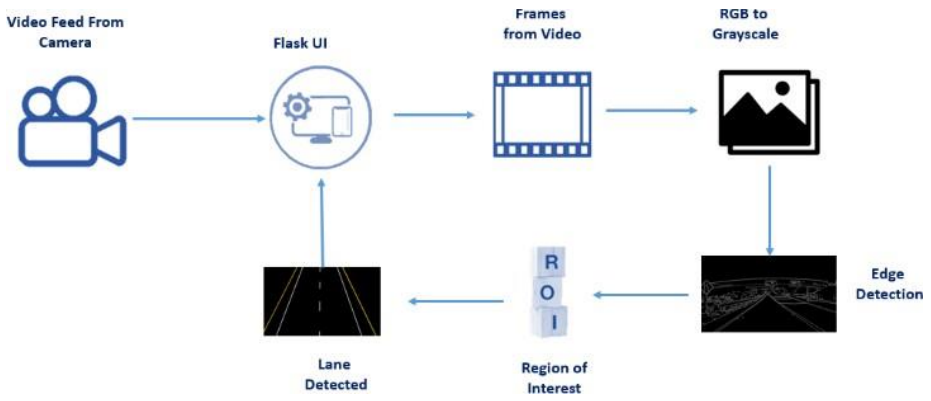
IV. PROPOSED SYSTEM

To overcome the drawbacks of the existing system, the proposed system has been evolved. This project aims to detect the lane for self-driving cars to avoid accidents and also to increase the lane capacity by using latest techniques in the field of Artificial Intelligence.

Advantages

- Reduce accidents.
- Increase Lane capacity.
- Lower fuel consumption.
- More effective and efficient parking.
- Man power consumption is reduced.
- The paper work is reduced in a way to provide license to the drivers.so that time consumption is reduced.

V. ACTUAL WORK



OpenCV Python Packages and Initializations:

- We will be Setting up things for Lane Detection, for example, required packages, declaring variables for global use, Gathering Data used to process, initializing a flask app to integrate our predicted output into web applications.

Import Necessary Packages:

- There are a set of packages that would be essential in order to use functions in this project. Here, we are going to import and make them all ready to use.

Intilazing Global Variables:

- Some variables need initialization before stepping into the mainframe of OpenCV Coding. Here Those variables and calculations are made accordingly.

Define Flask App and Load Data:

- The following simple snippet implements the corresponding interface and also supports whatever Flask has to offer us on top of that. It's essential to use any flask applications.

app= Flask(_name_)

- In this activity, we get ready with video input needed to process the lane detection.
- In OpenCV, video is captured using Video Capture () Function. It takes one parameter – the source of the video. It can either be from the External Camera (Front or Rear Camera) or any previous recorded Video.
  Provide 0 for Front Camera and 1 for Rear Camera. If you want to use pre-recorded video, Provide the path of the source video.

Image Manipulation:

- After Loading of Data, it has to undergo few transformations in order to make it simpler and efficient while performing further tasks. We follow open cv functions to make it possible.

Process Image Function:

- In this activity, we start using functions that focus on Image Conversion to grayscale and Region of Interest, here it is White Lanes. They will be highlighted and stored in a variable.

- matrix: cv2.getPerspectiveTransform
- minv : cv2.getPerspectiveTransform

Wrapping up the necessary points according to the perspective Warp.

- Get Birdseye by using cv2.warpPerspective function.
- Get and store the birdseye window dimensions to set limits for lanes.
- Divide the birdseye view into two halves to separate left & right lanes by using the slicing concept of NumPy.

Visualization:

- Visualization is used to enhance the picturising the data after every step. That way we will have a pictorial or graphical understanding of data as well as working procedure.

Histogram:

- A histogram is an approximate representation of the distribution of numerical data.

Visualizing:

- plot Histogram from Input .
- x-coordinates of left & right lanes to calculate lane width in pixels.
- Return histogram and x-coordinates of left & right lanes to calculate lane width in pixels.
-

Feature Extraction:

- In Feature Extraction we are going to extract the features based on our insights provided by some open CV functions.

Slide Window Search:

Slide_window_search: In this Activity, we Store all the Data into Stack, With the values obtained with the help of bird's Eye view and Histogram.

- Find the starting coordinates of the left and right lane lines in the frame.
- We do that by, Storing images into out_img variable as a stack using np.dstack function of NumPy.
- We are going to use Loops to iterate through frames in the video and search for lane points.
- It is followed by setting and checking margins and indices of lanes

- We get coordinates of Left and Right Lanes from the Wrap perspective.
- Now, Get x and y values by using "np.polyfit" function .
- Truncate x and y values using np.trunc function into the same values.
- Finally, show exact x and y values obtained after search.

General Search:

In this general search activity, we use a few other similar techniques to extract the Lane Features and make it easy to measure the distances.

- load Binary Wrapped non-zero elements:
- Load Non- zero indices and add margins
- Refit x and y values
- Apply cv2.fillPoly again
- result: cv2.addWeighted

Here we are going to check the status of Data in between perpetual change it is undergoing.

Measure Lane Curvature and Off Center:

Now we have to start measuring lane details like curvature, indices, off center values and more. To make the prediction accurate. That's what we will be going to do in this activity.

1. Measure_lane_curvature :

- Reverse to match top-to-bottom values of X and Y.
- Choose the maximum y-value, corresponding to the bottom of the image.
- Fit new polynomials to x, y in world space.
- Check for new input's radii.
- Decide if it is a left or a right curve.

2. Off Center:

- Calculating deviation in meters using Mean Values.
- Pixel deviation using Input shape and Absolute Value.

Draw lane Lines and Add Text:

1. Draw Lane Lines

- Drawing info of left, right, left_fit, right_fit and ploty function.
- Add Wrap Perspective and weights for Original Image.

2. Add Text

- Add the radius and center position to the image.
- Show Radius, Direction, Off-center.

Flask and Application Building:

- Here, we will be building a web application that is integrated to the model we built. A UI is provided for the uses where he has to enter the values for predictions. The enter values are given to the saved model and prediction is showcased on the UI.

Integrate Web Pages and Model:

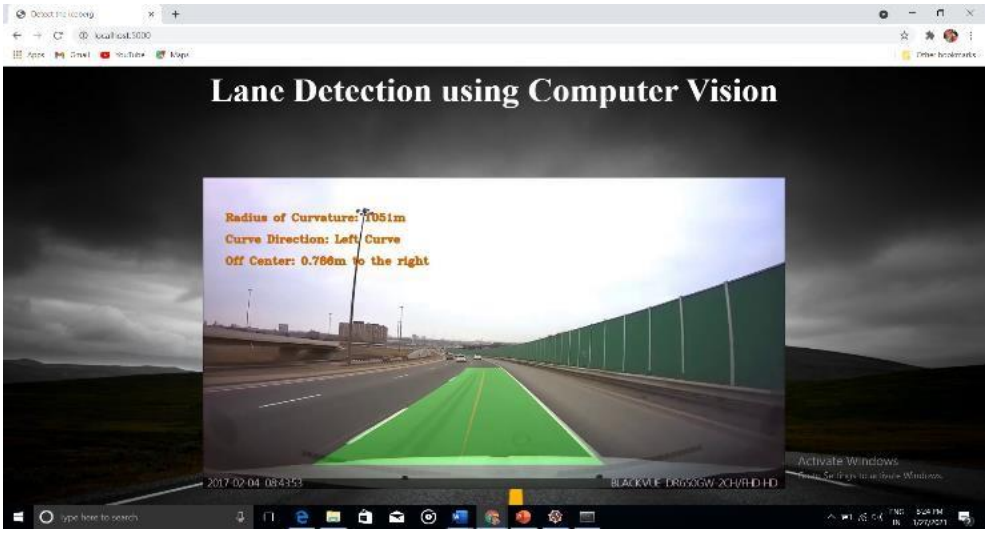
Render HTML Page:

- Here we are routing our app to function. This function retrieves all the values from the HTML page using Post request. We are requesting a text from the home .html using the request function. That is stored in an array. This array is passed to the model. Predict function (). This function returns the prediction. And this prediction value is rendered to the text that we have mentioned in the prediction.html page earlier.

Rendering Template:

- In gen (), Following Functions are called: (perspectiveWarp , processImage, plotHistogram, slide_window_search general_search, measure_lane_curvature, draw_lane_lines directionDeviation, AddText, cv2.VideoWriter)
- Return Response and call the main function of flask

VI. RESULTS



VII. CONCLUSION

In the methodology, we made use of the OpenCV library and its functions such as the Canny Function through which we achieved edge detection. Then we prepared a mask of zero intensity and mapped our region of interest by performing the bitwise operation. Then we used Hough Transform technique that detected the straight lines in the image and identified the lane lines. We made use of the polar coordinates since the Cartesian coordinates don't give us an appropriate slope of vertical and horizontal lines. Finally, we combined the lane image with our zero-intensity image to show lane lines.

REFERENCES

[1]. Aditya Singh Rathore (2019). Lane Detection for Autonomous Vehicles using OpenCV Library.

[2]. Haque, Md & Islam, Md & Alam, Kazi & Iqbal, Hasib & Shaik, Md. (2019). A Computer Vision based Lane Detection Approach. International Journal of Image, Graphics and Signal Processing. 11. 27-34. 10.5815/ijigsp.2019.03.04

[3]. Abhishek Goyal, Mridula Singh, Anand Srivastava (2019). Lane Detection on Roads using Computer Vision.

[4]. Punagin, Akash. (2020). Analysis of Lane Detection Techniques on Structured Roads using OpenCV. International Journal for Research in Applied Science and Engineering Technology. 8. 2994-3003. 10.22214/ijraset.2020.5502.

[5]. Mrs.M.Ambika, T.Vinothini, D.Rakshanaa(2019). Lane Deviation Warning System (Advanced Driver Assistance).

