Self Driving Lane Detection Car Using Python and Opencv on Raspberry Pi

Venkata Shiva Prasad Nannuri, Sai Santosh Kumar Mantha, Nikhilesh Pottipally, Sai Krishna Kodati and Suresh T.V Kumar

February 25, 2021

SELF DRIVING LANE DETECTION CAR USING PYTHON AND OPENCV ON RASPBERRY PI

Venkata Shiva Prasad Reddy
Electronics and communication Engineering
Bharat Institute of Engineering and Technology, Hyderabad, India.
spr81189@gmail.com

Sai Santosh Kumar
Electronics and communication Engineering
Bharat Institute of Engineering and Technology, Hyderabad, India.
saisantosh6190@gmail.com

P.Nikhilesh
Electronics and communication Engineering
Bharat Institute of Engineering and Technology, Hyderabad, India.
nikhileshpottipally21@gmail.com

K. Sai Krishna
Electronics and communication Engineering
Bharat Institute of Engineering and Technology, Hyderabad, India.
saikrishnakodati11@gmail.com

T.V. Suresh Kumar (Guide)
Assistant Professor
Electronics and communication Engineering
Bharat Institute of Engineering and Technology, Hyderabad, India.
Tvsuresh@biet.ac.in

Abstract

In this study, we present a perception algorithm that is based purely on vision or camera data. We focus on demonstrating a powerful end-to-end lane detection method using contemporary computer vision techniques for self-driving cars. We first present a minimalistic approach based on edge detection and polynomial regression which is the baseline approach for detecting only the straight lane lines. We then propose an improved lane detection technique based on perspective transformations and histogram analysis. In this latter technique, both straight and curved lane lines can be detected. To demonstrate the superiority of the proposed lane detection approach over the conventional approach, simulationresults in different environments are presented.

Keywords:

Raspberry pi ,Lane detection, Hough Transform, Threshold, Python Language , OpenCV Library.

Introduction

Driver Assistant System is designed to assist drivers in the perception of any dangerous situations before, to avoid accidents after sensing and understanding the environment around itself. To date there have been numerous studies into the recognition. Traffic accidents have become one of the most serious problems. The reason is that most accidents happen due to negligence of the driver. Rash and negligent driving could push other drivers and passengers in danger on the roads. More and more accidents can be avoided if such dangerous driving condition is detected early and warned to other drivers. Most of the roads, cameras and speed sensors are used for monitoring and identifying drivers who exceed the permissible speed limit on roads and motorways. This simplistic approach, and there are no restrictions.

Algorithm

The block diagram of a proposed lane detection system on Raspberry Pi is shown in the figure below:

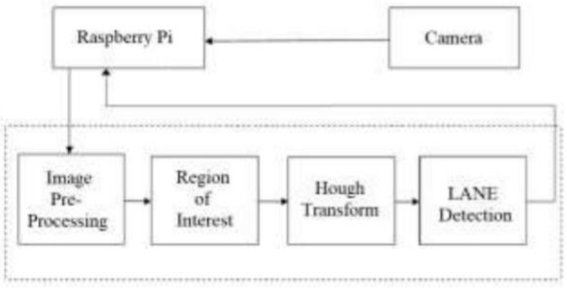


Figure 1: Block diagram of Lane Detection System Each blocks of a block diagram are explained one by one below:
1. Capture input image: Hardware like Camera is used to take input image.
2. Image Preprocessing: To enhance the quality of image, we need to preprocessit.The processes like noise reduction, edge detection, contrast and color management are applied.
3. Region of interest: In determining the computational complexity of lane identification and LDI system, ROI plays an
important role to detect it. Here only the selected are as is detected or taken for the next level of processing. These selected ROI images are then used for lane detection using a proposed algorithm. The selection of ROI reduces the processing time of the frames.
4. Hough Transform: The Hough Transform is implemented on images after the canny edges detection has taken place so as to obtain the image pixels that are desired ones. So here in this system to detect the lanes marking from the image data, Hough Transform is used.
5. Lane Detection: Here, the Lane will be marked with a separate color. Two important algorithms Canny Edge Detection and Hough Transform are used to implement Lane Detection System which are explained below:

Canny edge detection

Canny edge detection is a technique to extract useful structural information from different vision objects and dramatically reduce the amount of data to be processed. It has been widely applied in various computer vision systems. Canny has found that the requirements for the application of edge detection on diverse vision systems are relatively similar. Thus, an edge detection solution to address these requirements can be implemented in a wide range of situations. The general criteria for edge detection include: [1]
1. Detection of edge with low error rate, which means that the detection should accurately catch as many edges shown in the image as possible.
2. The edge point detected from the operator should accurately localize on the center of the edge.
3. A given edge in the image should only be marked once, and where possible, image noise should not create false edges.

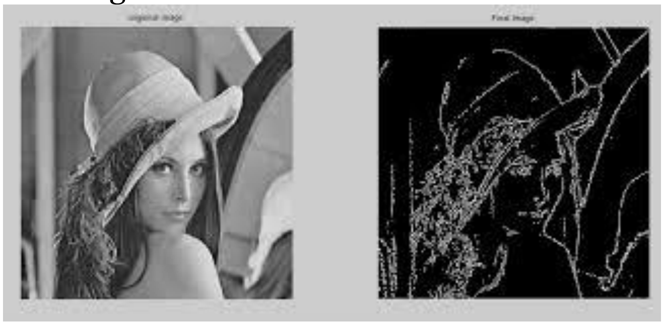
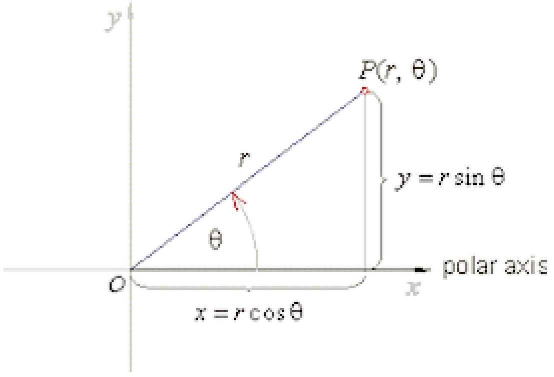


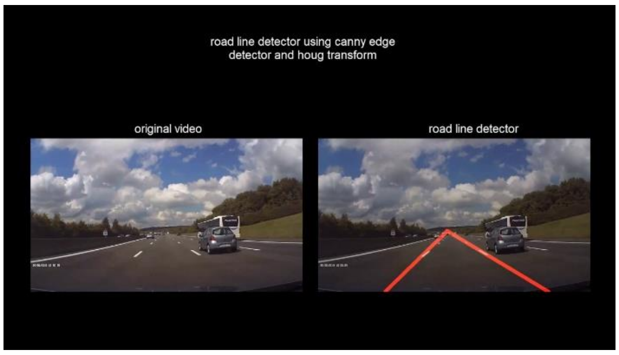
Figure2: Example for canny edge Detection [1]

Hough Transform

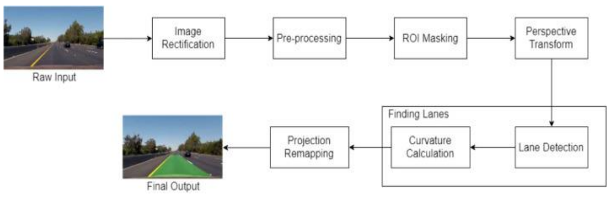
The features of various shape inside an image can be separated using a technique called as Hough Transform. This technique is generally used for the identification of arbitrary shapes such as straight lines, circles, ellipses, etc. The Hough Transform is implemented on images after the canny edges detection has taken place so as to obtain the image pixels that are desired ones. So here in this system to detect the lanes marking from the image data, Hough Transform is used. The primary point of interest of theHough Transform technique allows gaps in feature boundary descriptions and is not affected by noise. In general, the straight-line equation is given by y = mx + c can be represented as point (c, m) in the parameter space or Cartesian co-ordinate system. For the Hough transform we convert the equation to polar co- ordinates i.e. in terms of rho and theta. [3]



Where r is the distance from the initial position to closest point on straight line and theta (ɵ) is the angle between the line connecting the origin and the x axis. The (r, ɵ) plane is referred to as Hough space. The Hough transform detect the straight line in two dimensional arrays (matrix). Each element of the matrix has values equal to the sum of the pixels that are positioned on line. So, the elements with the highest values represent the straight lines.

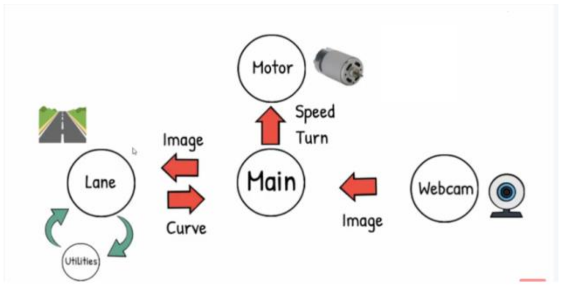

road line detector using canny edge detector and houg transform
original video
road line detector

Proposed Lane Detection System

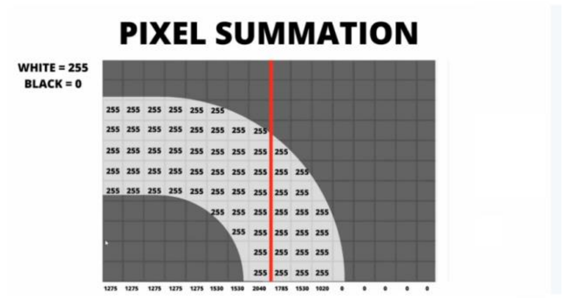


We are using the concept of modularity. Modularity is that we have separate files for each of the tasks to perform . so we have one main module that will handle all other files . so each file we call it a different module . so if you we want to use a ps4 , for that we will create a module for that and we will connect it to main module . the same way if you want to run the motors we have a motor module . in the same way we have lane detection and tracking module . these all are seperate modules that we can add and remove from our robot . so the advantage is that the coding becomes very neat and becomes very strategic where you can add and remove different modules without changing a lot of stuff and you can use this modules in any other projects also .

In this project we are having a main module that will first of all connect to webcam module . The images captured in the webcam module are send to main module and are used by lane module to detect the lanes from the image and sends us back the curve values on how much we need to turn the wheels of the robot . we also have a utility file that is linked to lane module . The reason for this is we don’t want to write all the code in one module if it is too long . we will write all the supporting functions in the utility file and we can relate to lane module. Once the curve is reached the main module , we can send this curve to motor module which will turn the motors based on the speed we have provided or based on the turn we have provided or received from the lane module .



We are going to use pixel summation method. The idea is that we are summing up the pixels in a column. Black is basically 0 and white is 255. Now because we are using 8 bit integers, we have 2 power 8 values that are from 0 to 256 values . We gave white as 255 and black as 0.

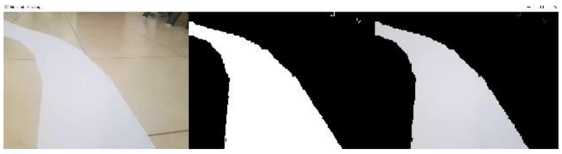


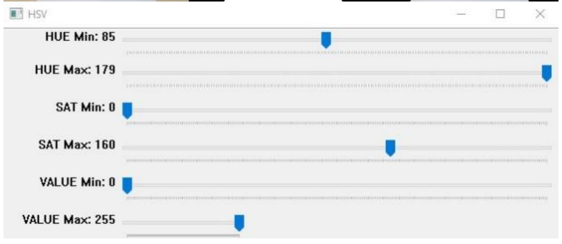
so with summing up the pixels in the column and with little bit of math we can not only determine whether we have to turn left or right or go straight but also how much we have to turn left or right . so that is the value of the curve . This is not a complicated process and we are not using fancy algorithms . It will be a easy process and we will go step by step and we will see the output of each steps .

Eventhough we are going to run our code in raspberry pi we are going to write our code in pc and the reason for that is it makes the coding process much easier and its much faster to debug . And once the module is prepared we can add it to our existing module and run it .

STEP 1 – Thresholding

Now the idea here is to get the path using either Color or Edge Detection. In our case we are using regular A4 White paper as path, therefore we can simply use the color detection to find the path. We will write the Thresholding function in the Utilities file.

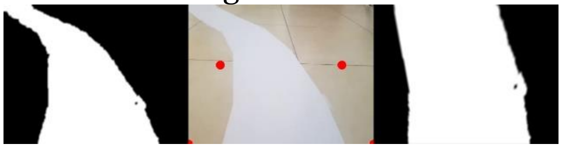




HUE Min: 85
HUE Max: 179
SAT Min: 0
SAT Max: 160
VALUE Min: 0
VALUE Max: 255

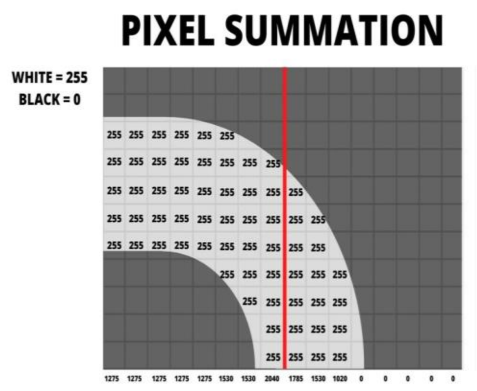
STEP 2 – WARPING

We don’t want to process the whole image because we just want to know the curve on the path right now and not a few seconds ahead. So we can simply crop our image, but this is not enough since we want to look at the road as if we were watching from the top . This is known as a bird eye view and it is important because it will allow us to easily find the curve. To warp the image we need to define the initial points. These points we can determine manually. So to make this process easier we could use track bars to experiment with different values. The idea is to get a rectangle shape when the road is straight.

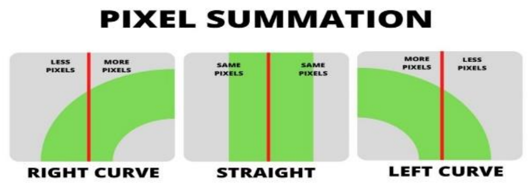


STEP 3-Histogram

Now comes the most important part, finding the curve in our path . To do this we will use the summation of pixels. But what is that? Given that our Warped image is now binary i.e it has either black or white pixels, we can sum the pixel values in the y direction. Lets look at this in more detail.



The picture above shows all the white pixels with 255 value and all the black with 0. Now if we sum the pixels in first column it yeilds 255+255+255+255+255 = 1275. We apply this method to each of the columns. In our original Image we have 480 pixels in the width. Therefore we will have 480 values. After summation we can look at how many values are above a certain threshold hold lets say 1000 on each side of the center red line. In the above example we have 8 columns on the left and 3 columns on the right. This tells us that the curve is towards left. This is the basic concept, though we will add a few more things to improve this and get consistent results. But if we look deeper into this we will face a problem. Lets have a look.



The above images shows the 3 cases where this methods would work. We can clearly see that when the curve is right the number of pixels on the right hand side are more than the left and vise versa. And when its straight the number of pixels are approximately same on both sides.

STEP 4 – Averaging

Once we have the curve value we will append it in a list so that we can average this value. Averaging will allow smooth motion and will avoid any dramatic movements.

STEP 5 – Display

Now we can add options to display the final result . We will add an input argument to our main ‘getLaneCurve’ function so that we can have the flexibility of turning it on and off, since raspberry pi would run at very slow speeds if we display and run at the same time.

Conclusion

The existing system which uses canny edge detection and hough transform usually detects straight lines. But it becomes very hard if there is a curve. When a curve appears the existing system usually cannot detect the lane accurately and there is a chance of getting hit by other vehicle. we have to turn the car according to the curve.

Our proposed system uses simple methods like thresholding, pixel summation methods and histogram analysis to detect the curves . This method is very much accurate when compared to the existing system. it detects the curve accurate enough to turn the motor in the provided speed.

References

1. Mr.Mustafa Surti, # Prof. (Dr.) Bharati Chourasia, “Real time lane detection system using Python and OpenCV on Raspberry Pi”, International Journal for Research in Engineering Application & Management (IJREAM) ISSN : 2454-9150 Vol-05, Issue-06, Sep 2019 .

2. Abdulhakam.AM.Assidiq, Othman O. Khalifa, Md. Rafiqul Islam, Sheroz Khan, “Real Time Lane Detection for Autonomous Vehicles “, Proceedings of the International Conference on Computer and Communication Engineering 2008 , May 13-15, 2008 Kuala Lumpur, Malaysia .