1. INTRODUCTION

1.1 INTRODUCTION

The traffic safety becomes more and more convincing with the increasing urban traffic. Exiting the lane without following proper rules is the root cause of most of the accidents on the avenues. Most of these are result of the interrupted and lethargic attitude of the driver. Lane discipline is crucial to road safety for drivers and pedestrians alike. The system has an objective to identify the lane marks. It's intent is to obtain a secure environment and improved traffic surroundings. The functions of the proposed system can range from displaying road line positions to the driving person on any exterior display, to more convoluted applications like detecting switching of the lanes in the near future so that one can prevent concussions caused on the highways. Actuate detection of lane roads is a critical issue in lane detection and departure warning systems. If an automobile crosses a lane confinement then vehicles enabled with predicting lane borders system directs the vehicles to prevent collisions and generates an alarming condition. These kind of intelligent system always makes the safe travel but it is not always necessary that lane boundaries are clearly noticeable, as poor road conditions inadequate quantity of paint used for marking the lane boundaries makes it hard for system to detect the lanes with accuracy and other reasons can include environmental effects like shadows from things like trees or other automobiles, or street lights, day and night time conditions, or fog occurs because of invariant lightening conditions. These factors causes problem to distinguish a road lane in the backdrop of a captured image for a person. In order to deal with above stated problems arising due to changes in lane boundaries. The algorithm followed in this paper is to detect lane markings on the road by giving the video of the road as an input to the system by using computer vision technology and primarily designed with the objective of reducing the frequency of accidents. System can be installed in cars and taxis in order to prevent the occurrence of accidents due to reckless driving on the roads. In school buses as it will guarantee the safety of the children. Moreover, performance of the driver can also be monitored, Road Transportation Offices can use the setup to check and report the negligence of drivers and lack of attention on the roads.

1.2 EXISTING SYSTEM

The traffic safety becomes more and more convincing with the increasing urban traffic. Exiting the lane without following proper rules is the root cause of most of the accidents on the avenues. Most of these are result of the interrupted and lethargic attitude of the driver. Lane discipline is crucial to road safety for drivers and pedestrians alike. The system has an objective to identify the lane marks. It's intent is to obtain a secure environment and improved traffic surroundings. The functions of the proposed system can range from displaying road line positions to the driving person on any exterior display, to more convoluted applications like detecting switching of the lanes in the near future so that one can prevent concussions caused on the highways. Actuate detection of lane roads is a critical issue in lane detection and departure warning systems. If an automobile crosses a lane confinement then vehicles enabled with predicting lane borders system directs the vehicles to prevent collisions and generates an alarming condition. These kind of intelligent system always makes the safe travel but it is not always necessary that lane boundaries are clearly noticeable, as poor road conditions

1.3 PROPOSED SYSTEM

n road-transport terminology, a lane departure warning system (LDWS) is a mechanism designed to warn the driver when the vehicle begins to move out of its lane (unless a turn signal is on in that direction) on freeways and arterial roads. These systems are designed to minimize accidents by addressing the main causes of collisions: driver error, distractions and drowsiness. In 2009 the U.S. National Highway Traffic Safety Administration (NHTSA) began studying whether to mandate lane departure warning systems and frontal collision warning systems on automobiles.

There are three types of systems:

Systems which warn the driver if the vehicle is leaving its lane with visual, audible, and/or vibration warnings (lane departure warning, LDW)

Systems which warn the driver and, with no response, automatically take steps to ensure the vehicle stays in its lane (lane keeping assist, LKA/LKS)

Systems which assist in oversteering, keeping the car centered in the lane, and asking the driver to take over in challenging situations (lane centering assist, LCA)

1.4 WORKING PRINCIPLE

A lane detection system used behind the lane departure warning system uses the principle of Hough transform and Canny edge detector to detect lane lines from realtime camera images fed from the front-end camera of the automobile. A basic flowchart of how a lane detection algorithm works to help lane departure warning is shown in the figures.

Lane keeping assist is a feature that, in addition to the lane departure warning system, automatically takes steps to ensure the vehicle stays in its lane. Some vehicles combine adaptive cruise control with lane keeping systems to provide additional safety.

While the combination of these features creates a semi-autonomous vehicle[non sequitur], most require the driver to remain in control of the vehicle while it is in use. This is because of the limitations associated with the lane-keeping feature.

The lane keeping assist system is being achieved in modern vehicle systems using image processing techniques called hough transform and canny edge detection techniques. These advanced image processing techniques derive lane data from forward facing cameras attached to the front of the vehicle. Real-time image processing using powerful computers like Nvidia's Drive PX1 are being used by many vehicle OEMs to achieve fully autonomous vehicles in which Lane detection algorithm plays a key part. Advanced lane detection algorithms are also being developed using deep learning and neural network techniques.[31] Nvidia has achieved high accuracy in developing self-driving features including lane keeping using the neural network based training mechanism in which they use a front facing camera in a car and run it through a route and then uses the steering input and camera images of the road fed into the neural network and make it 'learn'. The neural network then will be able to change the steering angle based on the lane change on the road and keep the car in the middle of the lane.

A lane keeping assist mechanism can either reactively turn a vehicle back into the lane if it starts to leave or proactively keep the vehicle in the center of the lane. Vehicle companies

often use the term "lane keep(ing) assist" to refer to both reactive lane keep assist (LKA) and proactive lane centering assist (LCA) but the terms are beginning to be differentiated.

Tesla uses the most advanced lane assist system (kind of LKA) combined with their adaptive cruise control system marketed together as 'Autopilot'.

In 2020, UNECE released an automated lane keeping system regulation which include features such as lane-keeping and adaptative speed for specific roads up to 60 km/h.

Tesla includes features like lane-keeping assist and also automatic lane changing without driver input. A similar technology to lane assist is used to do autopark feature as well.

2.LITERATURE SURVEY

Lane Detection for Autonomous Car via Video Segmentation There are many steps in detecting lanes on a road, first comes the camera calibration. Cameras uses curved lenses to form an image, and light rays often bend a little too much or too little at the edges of these lenses. Images can be undistorted mapping distorted points to undistorted points such as a chessboard. This distortion correction is then applied to raw images, convert images to grayscale, apply gradients and finally apply deep leaning. Then perspective transform is applied to the binary image from bird's eye view. A Global Convolution Networks (GCN) model is used to rectify the classification and localization issues for semantic segmentation of lane. The model is evaluated using colour-based segmentation. For classification task the conical CNN have proved to be better. For object segmentation, the size of the kernel matters a lot. But a larger size results in increase in weights and the number of parameters also increases. In the model 1 D kernels were used to increase performance with less weights. The dataset was collected from Carla Simulator which contained 3000 images of road. The sky portion and car hood part from the images was cropped to increase the performance. [1] [International Research Journal of Engineering and Technology] Prof A. B. Deshmukh e t a [2] in their paper that Lane detection is an essential component of Advance Driver Assistance system (ADAS). Many different approaches have been proposed till today by researchers but still it is a challenging task to correctly detect the road lanes in various environmental conditions. The main purpose of the system is to detect the lane departure to avoid road accidents and to provide safety for pedestrians. The proposed method detects the road edges using the canny edges detector whereas the feature extraction technique like Hough transform is used in image analysis and digital signal processing. The main input to the system is camera captured images in order to detect and track the road boundaries. This concept of image processing is implemented using OpenCV library function on Raspberry pi hardware. This method can correctly detect the roads in various challenging situations. Results shows that the proposed method can detect both the straight and curves lanes correctly. [2] Recently, many studies are conducted based on advanced driver assistance system (ADAS) to avoid car accidents. Mostly, the lane departure warning system (LDWS)system warns the driver when the vehicles tend to depart of its lane, which is the most basic and important part of the ADAS.

2.1 BLOCK DIAGRAM OF LANE DETECTION

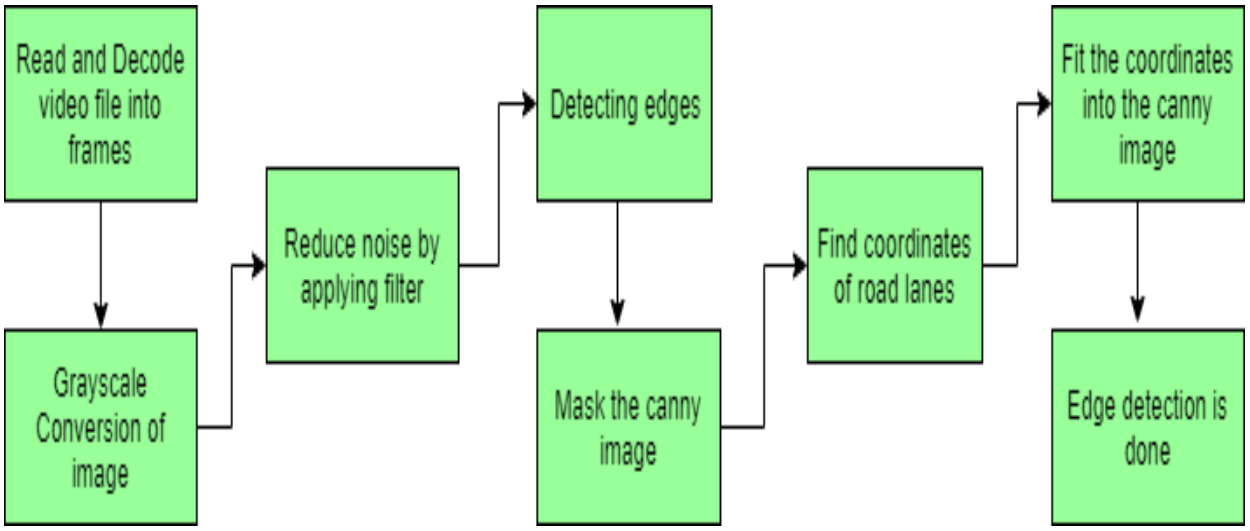


Figure 1. block diagram of lane detection steps

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 preprocess it.

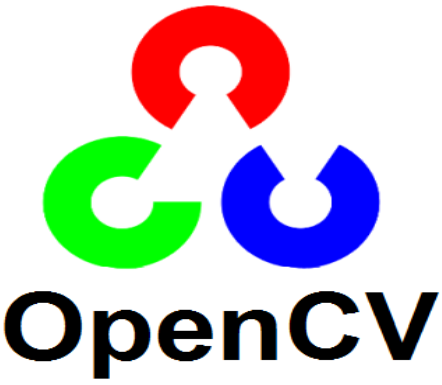
The processes like noise reduction, edge detection, contrast and color management are applied.

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

2.2 OPEN CV



OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code.

The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc. OpenCV has more than 47 thousand people of user community and estimated number of downloads exceeding 18 million. The library is used extensively in companies, research groups and by governmental bodies.

Along with well-established companies like Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, Toyota that employ the library, there are many startups such as Applied Minds, VideoSurf, and Zeitera, that make extensive use of OpenCV. OpenCV's deployed uses span the range from stitching streetview images together, detecting intrusions in surveillance video in Israel, monitoring mine equipment in China, helping robots navigate and pick up objects at Willow Garage, detection of swimming pool drowning accidents in Europe, running interactive art in Spain and New York, checking runways for debris in Turkey, inspecting labels on products in factories around the world on to rapid face detection in Japan.

It has C++, Python, Java and MATLAB interfaces and supports Windows, Linux, Android and Mac

OS. OpenCV leans mostly towards real-time vision applications and takes advantage of MMX and SSE instructions when available. A full-featured CUDAand OpenCL interfaces are being actively developed right now. There are over 500 algorithms and about 10 times as many functions that compose or support those algorithms. OpenCV is written natively in C++ and has a templated interface that works seamlessly with STL containers

3.SYSTEM ANALYSIS

3.1 SOFTWARE REQUIREMENTS:

Platform – Windows 10 pro

Software – Anaconda – Jupyter Notebook

3.2HARDWARE REQUIREMENTS:

Processor – Intel® core™ i3 CPU M 380

@2.53GHz.RAM – 4.00GB

Keyboard - 101 keys

4.DOMAIN

4.1PYTHON:

Python is a general purpose, dynamic, high level, and interpreted programming language It supports Object Oriented programming approach to develop applications. It is simple and easy to learn and provides lots of high-level data structures. Python is easy to learn yet powerful and versatile scripting language, which makes it attractive for Application Development. Python's syntax and dynamic typing with its interpreted nature make it an ideal language for scripting and rapid application development. Python supports multiple programming pattern, including object-oriented, imperative, and functional or procedural programming styles. Python is not intended to work in a particular area, such as web programming. That is why it is known as multipurpose programming language because it can be used with web, enterprise, 3D CAD, etc. We don't need to use data types to declare variable because it is dynamically typed so we can write a=10 to assign an integer value in an integer variable. Python makes the development and debugging fast because there is no compilation step included in Python development and edit-test-debug cycle is very fast.

4.2 PYTHON'S FEATURE SET:

The factors that played an important role in molding the final form of the language and are given by

1. Easy to code: Python is very easy to code. Compared to other popular languages like Java and C++, it is easier to code in Python.
2. Easy to read: Being a high-level language, Python code is quite like English. Looking at it, you can tell what the code is supposed to do. Also, since it is dynamically-typed, it mandates indentation. This aids readability.
3. Expressive: Suppose we have two languages A and B, and all programs that can be made in A can be made in B using local transformations. However, there are some programs that can be made in B, but not in A, using local transformations. Then, B is said to be more expressive than A. Python provides us with a myriad of constructs that

help us focus on the solution rather than on the syntax.

4. Free and Open-Source: Firstly, Python is freely available. You can download it from the link. Secondly, it is open source. This means that its source code is available to the public. You can download it, change it, use it, and distribute it. This is called FLOSS (Free/Libre and Open Source Software.

5. High- Level: This means that as programmers, we don't need to remember the system architecture. Nor do we need to manage the memory. This makes it more programmer-friendly and is one of the key python features.

6. Portable: We can take one code and run it on any machine, there is no need to write different code for different machines. This makes Python a portable language.

7. Interpreted: If you are any familiar with languages like C++ or Java, you must first compile it, and then run it. But in Python, there is no need to compile it. Internally, its source code is converted into an immediate form called bytecode. So, all you need to do is to run your Python code without worrying about linking to libraries, and a few other things. By interpreted, we mean the source code is executed line by line, and not all at once. Because of this, it is easier to debug your code. Also, interpreting makes it just slightly slower than Java, but that does not matter compared to the benefits it has to offer.

8. Object-Oriented: A programming language that can model the real world is said to be object-oriented. It focuses on objects and combines data and functions. Contrarily, a procedure-oriented language revolves around functions, which are code that can be reused. Python supports both procedure-oriented and object-oriented programming whichis one of the key python features. It also supports multiple inheritance, unlike Java. A class is a blueprint for such an object. It is an abstract data type and holds no values.

9. Extensible: If needed, you can write some of your Python code in other languages like C++. This makes Python an extensible language, meaning that it can be extended to other languages.

10. Embeddable: We just saw that we can put code in other languages in our Python source code. However, it is also possible to put our Python code in a source code in a different language like C++s.This allows us to integrate scripting capabilities into our program of the other language.

11. Large Standard Library: Python downloads with a large library that you can use so you don't have to write your own code for every single thing. There are libraries for regular expressions, documentation-generation, unit-testing, web browsers, threading,

databases, CGI, email, image manipulation, and a lot of other functionality.

12. GUI Programming: It also supports graphical user interface.

13. Dynamically Typed Python is dynamically-typed. This means that the type for a value is decided at runtime, not in advance. This is why we don't need to specify the type of data while declaring it.

Anaconda Distribution

Anaconda Navigator is a free open source software distribution of the languages like Python and R for Machine learning and Data Science. The aim is to simplifythe package management and deployment. The Anaconda distribution is used by over 6 million users, and it includes more than 250 popular data science packages suitable for Windows, Linux, and MacOS. Anaconda distribution comes with more than 1,000 data packages as well as the Conda package and virtual environment manager, called Anaconda Navigator [6], so it eliminates the need to learn to install each library independently.

Anaconda Navigator

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda distribution that allows users to launch applications and manage conda packages, environments and channels without using command-line commands. Navigator can search for packages on Anaconda Cloud or in a local Anaconda Repository, install them in an environment, run the packages and update them. It is available for Windows, macOS and Linux. Navigator is automatically included with Anaconda version 4.0.0 or higher.

Anaconda Cloud

Anaconda Cloud is a package management service by Anaconda where you can find, access, store and share public and private notebooks, environments, and conda and PyPI packages. Cloud hosts useful Python packages, notebooks and environments for a wide variety of applications. You do not need to log in or to have a Cloud account, to search for public packages, download and install them.You can build new packages using the Anaconda Client command line interface (CLI), then manually or automatically upload the packages to Cloud.

Conda

Conda is an open source package management system and environment management system that runs on Windows, macOS and Linux. Conda quickly installs, runs and updates packages and their dependencies. Conda easily creates, saves, loads and switches between

environments on your local computer. It was created for Python programs, but it can package and distribute software for any language.

Parts of Anaconda distribution

1. Jupyter Lab
2. Jupyter Notebook
3. QtConsole
4. Spyder
5. Glueviz
6. Orange
7. RStudio
8. Visual Studio Code

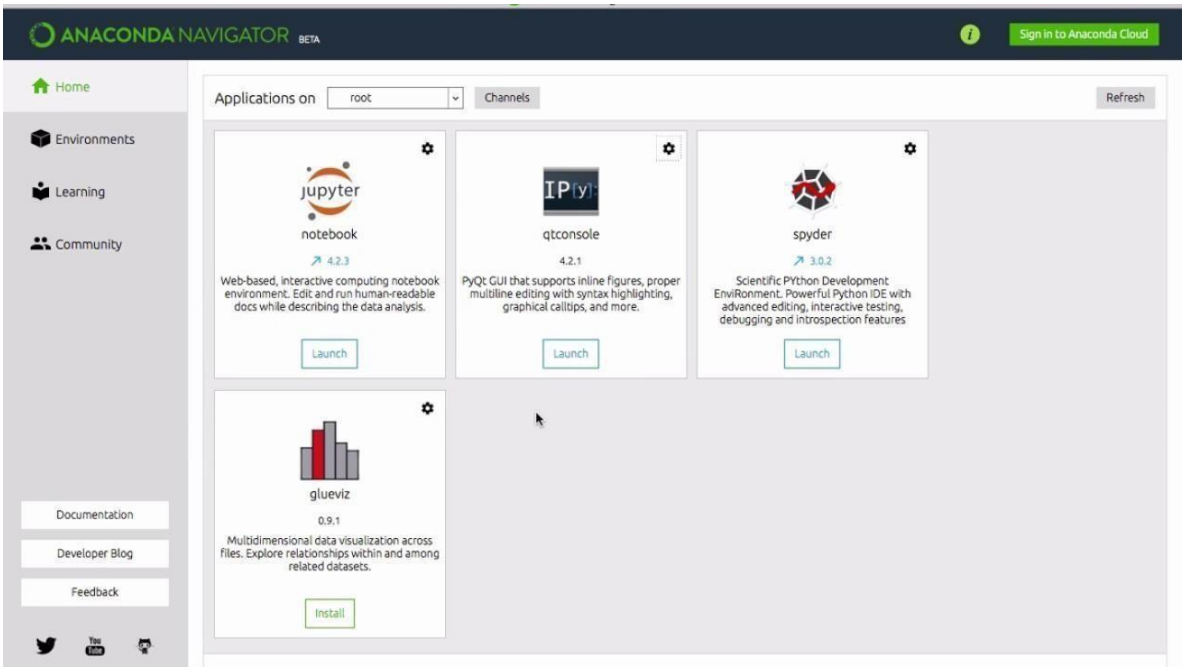


Figure 2. anaconda navigator

Jupyter

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

Language of choice

It supports more than 40 programming languages including Python, Spyder etc.

Share Notebooks

Notebooks can be shared with others using email, Dropbox, GitHub and the Jupyter Notebook Viewer.

Interactive Output

Your code can produce rich, interactive output: HTML, images, videos, LaTeX, and custom MIME types.

5. MODULE DESIGN

5.1.DATA COLLECTION :

The paper suggests collecting various images with shadows, multi-coloured paved surfaces and poorly defined borders. The images are from several sets of data collected from a camera mounted on vehicle. For training the algorithm, region of interest is selected manually that is labeled as road. A training set is created is which road region feature is labeled 1 and non-road region feature is labeled - 1. Then, the features are combined according to the combine function using correlation between them.

Figure 3. Image of roads

A CCD camera is fixed on the front-view mirror to capture the road scene. To simplify the problem, we assume that it is setup to make the baseline horizontal, which assures the horizon in the image, is parallel to the X-axis. Otherwise, we can adjust the image using the calibration data of the camera to

make it. Each laneboundary marking, usually a rectangle (or approximate) forms a pair of edge lines. In this paper, it was assumed that the input to the algorithm was a 620x480 RGB color image. Therefore the first thing the algorithm does is to convert the image to a grayscale image in order to minimize the processing time. Secondly, as presence of noise in the image will hinder the correct edge detection. Therefore, we apply (F.H.D.) algorithm [13] to make the edge detection more accurate. Then the edge detector is used to produce an edge image by using canny filter with automatic thresholding to obtain the edges, it will reduce the amount of learning data required by simplifying the image edges considerably. Then edged image sent to the line detector after detecting the edges which will produces a right and left lane boundary segment. The projected intersection of these two line segments is determined and is referred to as the horizon. The lane boundary scan uses the information in the edge image detected by the Hough transform to perform the scan. The scan returns a series of points on the right and left side. Finally pair of hyperbolas is fitted to these data points to represent the lane boundaries

5.1.1 IMAGE CAPTURING :

Image Capturing The input data is a color image sequence taken from a moving vehicle. A color camera is mounted inside the vehicle at the front-view mirror along the central line. It takes the images of the environment in front of the vehicle, including the road, vehicles on the road, roadside, and sometimes incident objects on the road. The on-board computer with image capturing card will capture the images in real time (up to 30 frames/second), and save them in the computer memory. The lane detection system reads the image sequence from the memory and starts processing. A typical scene of the road ahead is depicted by Figure 1. In order to obtain good estimates of lanes and improve the speed of the algorithm, the original image size was reduced to 620x480 pixels by Gaussian pyramid.

Figure 4 . Lanes of road Image

5.2 DATA PREPROCESSING :

Import the datasets and libraries—First step is usually importing the libraries that will be needed in the program. A library is essentially a collection of modules that can be called and used. Importing the dataset—Loading the data using cv2 library using cv2.VideoCapture() and cap.read () method.

```
cap = cv2.VideoCapture("test1.mp4")
while (cap.isOpened()):
    _, frame = cap.read()
    canny_image = canny(frame)
    cropped_canny = region_of_interest(canny_image)
    # cv2.imshow("cropped_canny",cropped_canny)
```

This data set can applying in different processing steps like decode video, Grayscale Image, Region of interest, Hough transformation, Canny Edge detection.etc

5.3 CONVERSION OF GRAYSCALE :

To retain the color information and segment the road from the lane boundaries using the color information this proved difficulties on edge detection and also it will effect the processing time. In practice the road surface can be made up of many different colors due to shadows, different pavement style or age, which causes the color of the road surface and lane markings to

change from one image region to another. Therefore, color image are converted into grayscale. However, the processing of grayscale images becomes minimal as compared to a color image. This function transforms a 24-bit, three-channel, color image to an 8- bit, single-channel grayscale image by forming a weighted sum of the Red component of the pixel value * 0.3 +Green component of the pixel value * 0.59 + Blue component for the pixel value *0.11 the output is the gray scale value for the corresponding pixel.

Each of the pixels for a grayscale image is described by a single number that describes the brightness of the pixel. In order to
smoothen an image, the typical answer would be to modify the value of a pixel with the average value of the pixel intensities
around it. Averaging out the pixels to reduce the noise will be done by a kernel. This kernel of normally distributed
numbers(np.array([[1,2,3],[4,5,6],[7,8,9]]) is run across our entire image and sets each pixel value equal to the weighted average
of its neighboring pixels, thus smoothening our image. In our case we will apply a 5x5 Gaussian kernel:
blur= cv2.GaussianBlur(gray,(5,5),0)
Below is the image with reduced noise:

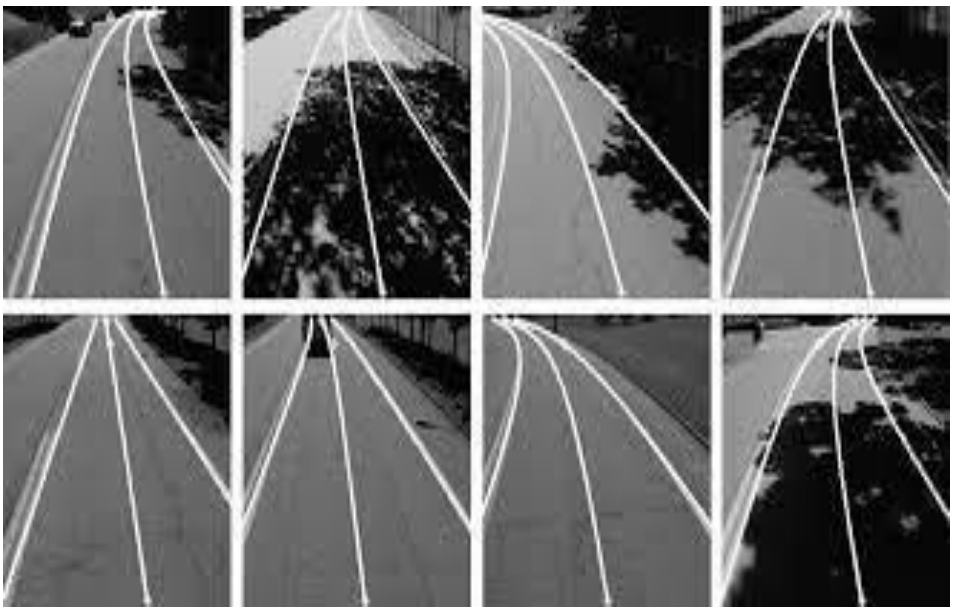


Figure 5. Guassian Blur Images

5.4 NOISE REDUCTION :

Noise is a real world problem for all systems and computer vision is no exception. The algorithms developed must either be noise tolerant or the noise must be eliminated. As presence of noise in our system will hinder the correct edge detection, so that noise removal is a pre requisite for efficient edge detection with the help of (F.H.D.) algorithm [13] that removes strong shadows from a single image. The basic idea is that a shadow has a distinguished boundary. Removing the shadow boundary from the image derivatives and reconstructing the image should remove the. A shadow edge image can be created by applying edge-detection on the invariant image and the original image, and selecting the edges that exist in the original image but not in the invariant image and to reconstruct the shadow free image by removing the edges from the original image using a pseudo-inverse filter.

5.5 EDGE DETECTION :

Lane boundaries are defined by sharp contrast between the road surface and painted lines or some type of non-pavement surface. These sharp contrasts are edges in the image. Therefore edge detectors are very important in determining the location of lane boundaries. It also reduces the amount of learning data required by simplifying the image considerably, if the outline of a road can be extracted from the image. The edge detector implemented for this algorithm and the one that produced the best edge images from all the edge detectors evaluated was the ‘canny’ edge detector [14].

canny = cv2.Canny(blur, 50, 150)

To find the maxima of the partial derivative of the image function I in the direction orthogonal to the edge direction, and to smooth the signal along the edge direction/Thus Canny's operator looks for the maxima of :

$$G_\sigma = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{x^2+y^2}{2\sigma^2} \right]$$

Where

$$n = \frac{\nabla G * I}{\left| \nabla G * I \right|}$$

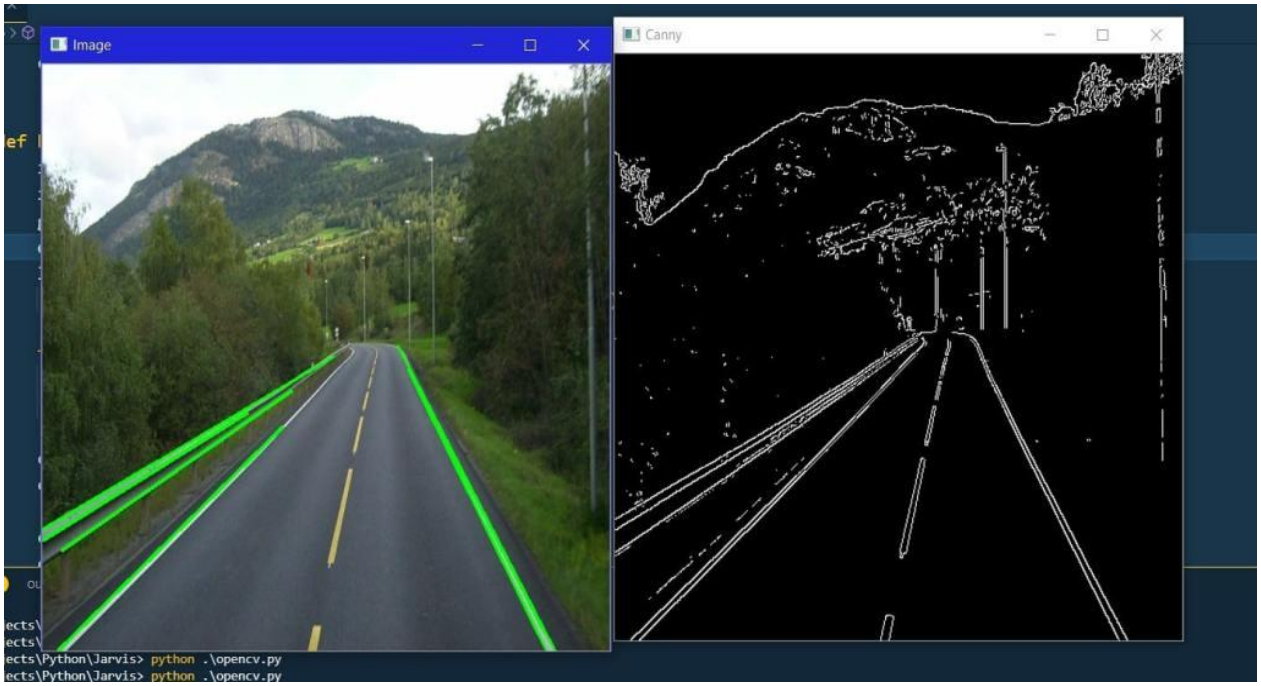


Figure 6.before and after canny Edge detector

It was important to have the edge detection algorithm that could be able to select thresholds automatically however, the automatic threshold that is used in the default Canny produced far too much edge information. Making a slight modification to the edge detected produced by canny has given more desirable results. The only changes necessary were to set the amount of non-edge pixels to the best value that gives more accurate edges in different conditions of image capturing environment. The canny edge detector also has a very desirable characteristic in that it does not produce noise like the other approaches.

5.6 REGION OF INTEREST:

The dimensions of the image are chosen which will contain the road lanes and mark it as our region of interest or the triangle. Then a mask is created which is same as the dimension of the image which would essentially be an array of all zeros. Now we fill the triangle dimension in this mask with the intensity of 255 so that our region of interest dimensions are white. Now I will do a bitwise AND operation with the canny image and the mask which will result in our final region of interest.

```
def region(image):
height= image.shape[0]
poly= np.array([[(200,height),(1100,height),(550,250)]])
mask= np.zeros_like(image)
 cv2.fillPoly(mask, poly,255)
 masked_image= cv2.bitwise_and(image,mask)
 return masked_image
```

Figure 7 . Image of mask

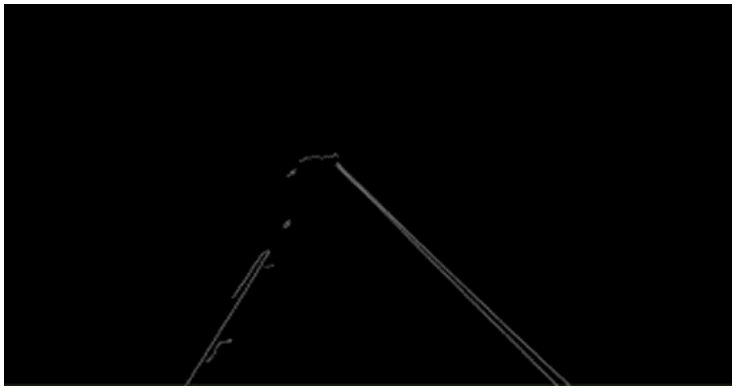


Figure 8 . Image of Canny Edges with Lanes

Figure 8. the image after doing a bitwise operation with the canny image and the mask, also called the masked_image:

5.7 HOUGH TRANSFARMATION :

Now we make use of hough transform technique that will detect straight lines in the image and thus identify the lane lines. We know that a straight line is represented by the below equation:

y= mx + b

And the slope of the line is simply a rise over run. If the y intercept and slope is given then the line can be plotted in the Hough Space as a single dot. There are many possible lines that can pass through this dot each line with different values for M and B. There are many possible lines that can cross each point individually, each line with different slope and y intercept values. However there is one line that is consistent with both points. We can determine that by looking at the point of intersection enough space because that point of intersection in Hough Space and that point of intersection represents the M and B values of a line consistent with crossing both the points. Now in order to identify the lines, we will first split our Hough space into a grid. Each bin inside the grid corresponding to the slope and y intercept value of the line. For every point of intersection in a Hough Space bin we're going to cast a vote inside of the bin that it belongs to. The bin with maximum number of votes will be our line. But as we know that the slope of a vertical line is infinity. So to express vertical lines, we will use polar coordinates instead of cartesian coordinates. So the equation of our line becomes:

roh= xcos(theta) + ysin(theta) def display_lines(image, lines):
line_image= np.zeros_like(image) if lines is not None:
for line in lines:
x1,y1,x2,y2 = line.reshape(4) cv2.line(line_image,(x1,y1),(x2,y2),(255,0,0),10)
return line_image
lines= cv2.HoughLinesP(cropped,2,np.pi/180, 100, np.array([]), minLineLength=40, maxLineGap=5)

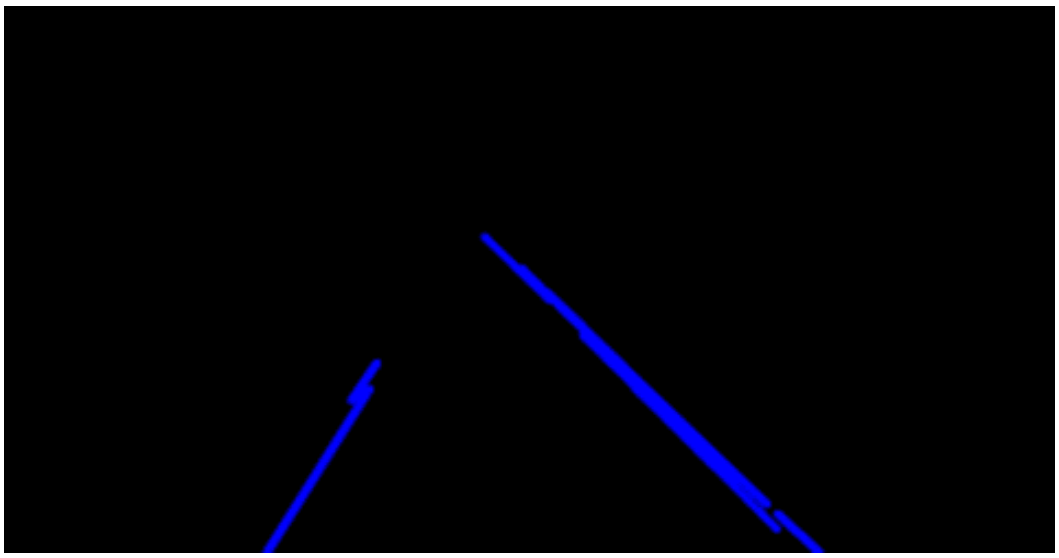


Figure 9. the image of the blue coloured lines drawn on a zero-intensity image:

Now we will combine our zero intensity image with our lane_image:

combo_image= cv2.addWeighted(lane_image,0.8,line_image, 1, 1)

Below is the combined image:

Figure 10. Image with Lane Detection

5.8 LANE BOUNDARY SCAN:

The lane boundary scan phase uses the edge image the Hough lines and the horizon line as input. The edge image is what is scanned and the edges are the data points it collects. The scan begins where the projected Hough lines intersect the image border at the bottom of the image. Once that intersection is found, it is considered the starting point for the left or right search, depending upon which intersection is at hand. From the starting point, the search begins a certain number of pixels towards the center of the lane and then proceeds to look for the first edge pixel until reaching a specified number of pixels after the maximum range. The range will be set to the location of the previously located edge pixel plus a buffer number of pixels further

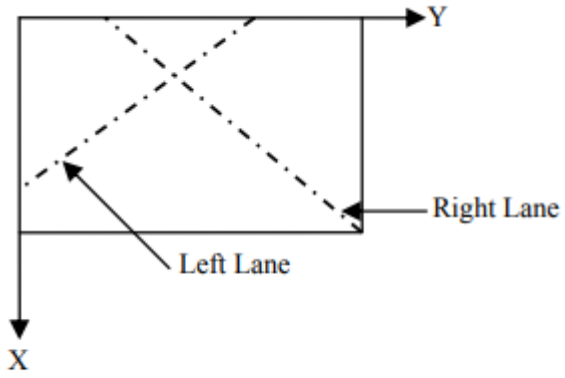


Figure 5. Image coordinates

Figure 11. Image of Coordinates

For the starting condition, the search will already be at the left- or right-most border, as shown in Figure 5 .so the maximum range will be the border itself. The extra buffer zone of the search will help facilitate the following of outward curves of the lane boundaries. The lane points are organized into two lists expressed as L(l) and L(r)

$$L^{(l)} = \{(u_1^{(l)}, v_1^{(l)}), (u_2^{(l)}, v_2^{(l)}), \ldots\ldots\ldots, (u_m^{(l)}, v_m^{(l)})\}$$
$$L^{(r)} = \{(u_1^{(r)}, v_1^{(r)}), (u_2^{(r)}, v_2^{(r)}), \ldots\ldots\ldots, (u_m^{(r)}, v_m^{(r)})\}$$

6. CODING

```
import cv2
import numpy as np


def canny(img):
    if img is None:
        cap.release()
        cv2.destroyAllWindows()
        exit()
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    kernel = 5
    blur = cv2.GaussianBlur(gray, (kernel, kernel), 0)
    canny = cv2.Canny(gray, 50, 150)
    return canny


def region_of_interest(canny):
    height = canny.shape[0]
    width = canny.shape[1]
    mask = np.zeros_like(canny)
    triangle = np.array([[
        (200, height),
        (800, 350),
        (1200, height), ]], np.int32)
    cv2.fillPoly(mask, triangle, 255)
    masked_image = cv2.bitwise_and(canny, mask)
    return masked_image
```

```
def houghLines(cropped_canny):
    return cv2.HoughLinesP(cropped_canny, 2, np.pi / 180, 100,
                           np.array([]), minLineLength=40, maxLineGap=5)


def addWeighted(frame, line_image):
    return cv2.addWeighted(frame, 0.8, line_image, 1, 1)


def display_lines(img, lines):
    line_image = np.zeros_like(img)
    if lines is not None:
        for line in lines:
            for x1, y1, x2, y2 in line:
                cv2.line(line_image, (x1, y1), (x2, y2), (0, 0, 255), 10)
    return line_image


def make_points(image, line):
    slope, intercept = line
    y1 = int(image.shape[0])
    y2 = int(y1 * 3.0 / 5)
    x1 = int((y1 - intercept) / slope)
    x2 = int((y2 - intercept) / slope)
    return [[x1, y1, x2, y2]]


def average_slope_intercept(image, lines):
    left_fit = []
    right_fit = []
    if lines is None:
        return None
    for line in lines:
```

```
        for x1, y1, x2, y2 in line:
            fit = np.polyfit((x1, x2), (y1, y2), 1)
            slope = fit[0]
            intercept = fit[1]
            if slope < 0:
                left_fit.append((slope, intercept))
            else:
                right_fit.append((slope, intercept))
    left_fit_average = np.average(left_fit, axis=0)
    right_fit_average = np.average(right_fit, axis=0)
    left_line = make_points(image, left_fit_average)
    right_line = make_points(image, right_fit_average)
    averaged_lines = [left_line, right_line]
    return averaged_lines


cap = cv2.VideoCapture("test1.mp4")
while (cap.isOpened()):
    _, frame = cap.read()
    canny_image = canny(frame)
    cropped_canny = region_of_interest(canny_image)

    lines = houghLines(cropped_canny)
    averaged_lines = average_slope_intercept(frame, lines)
    line_image = display_lines(frame, averaged_lines)
    combo_image = addWeighted(frame, line_image)
    cv2.imshow("result", combo_image)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

7. OUTPUT SCREENS

Input dataset

Fig 12.Input data set for lane detection

Output dataset

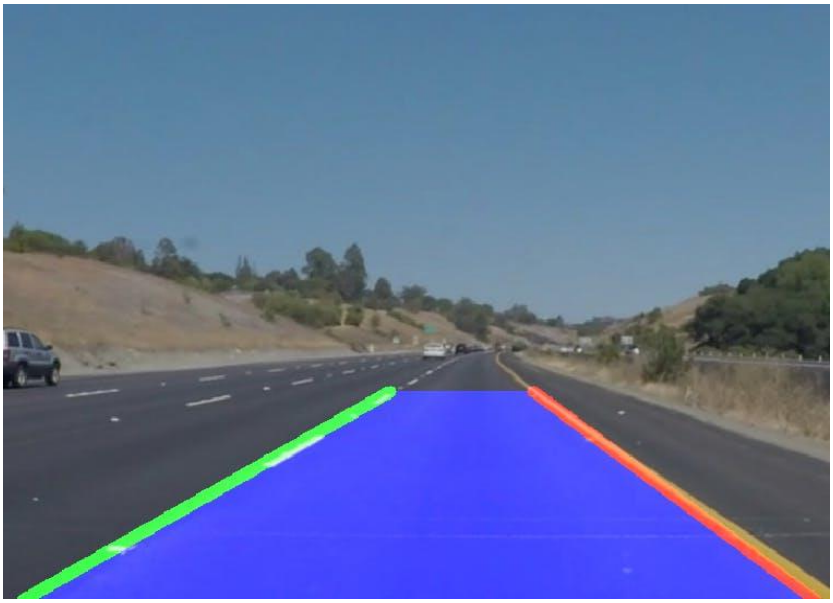


Figure 13. Image of Lane detection

8. CONCLUSION

In the procedure, we utilized the OpenCV library and its capacities, for example, the Canny Function through which we accomplished edge recognition. At that point we arranged a veil of zero force and planned our mapped our region of interest. At that point we utilized the Hough Transform strategy that recognized the straight lines in the picture and distinguished the path lines. We utilized the polar directions since the Cartesian directions don't give us a proper slant of vertical what's more, level lines. At last, we consolidated the path picture with our zero-intensity picture to show path lines. As far as path identification exactness and calculation tedious, the proposed path discovery calculation had clear favorable circumstances. It was helpful for extraordinarily upgrading the driving security of shrewd vehicles in the real driving conditions and viably meeting the continuous objective necessities of keen vehicles and assumed a significant part in clever vehicle driving help. Later on, the comprehensiveness and against mistake identification of the path recognition calculation can be additionally advanced and improved to abuse the general exhibition of the calculation. In this paper, a real time lane detection algorithm based on video sequences taken from a vehicle driving on highway was proposed. As mentioned above the system uses a series of images. Out of these series some of the different frames used are shown in the lane detection algorithm, (F.H.D.) algorithm conduct image segmentation and remove the shadow of the road. Since the lanes are normally long and smooth curves, we consider them as straight lines within a reasonable range for vehicle safety. The lanes were detected using Hough transformation with restricted search area. The proposed lane detection algorithm can be applied in both painted and unpainted road, as well as slightly curved and straight road as shown in Figure 9. There remained some problems in the lane detection due to shadowing and the Hough line and horizon overlaid, then in the lower left edge with the lane boundary points overlaid and in few cases lane scan fails to track the correct lane.

9. BIBILOGRAPHY

1. "International Research Journal of Engineering and Technology (IRJET)" Lane Detection for Autonomous Vehicles using OpenCV Library Aditya Singh Rathore B.Tech, J.K. Lakshmipat University.
2. "Abdulhakam.AM.Assidiq, Othman O. Khalifa, Md. Rafiqul Islam, Sheroz Khan Department of Electrical & Computer Faculty of Engineering,"- International Islamic University Malaysia.
3. https://towardsdatascience.com/data-preprocessing-concepts-fa946d11c825.
4. htttp://www.kaggle.com/c/walmart-recruiting-stores-sales-forecasting.
5. www.kdnuggets.com
6. http://www.themalaysian.blogspot.com/2006/08/fatal-roadaccidents-ranking-malaysia.html, August.2006.
7. http://www.itseurope/2007/ B. M, Broggi, "GOLD: A parallel real-time stereo Vision system for generic obstacle and lane detection", IEEE Transactions on Image Processing, 1998,pp. 4-6.
8. C. Kreucher, S. K. Lakshmanan, "A Driver warning System based on the LOIS Lane detection Algorithm", Proceeding of IEEE International Conference On Intelligent Vehicles. Stuttgart, Germany, 1998, pp. 17 -22.
9. Pomerleau D. and Jochem,"Rapidly Adapting Machine Vision for Automated Vehicle Steering", IEEE, 1996.
10. M. Chen., T. Jochem D. T. Pomerleau, "AURORA: A VisionBased Roadway Departure Warning System", In Proceeding of IEEE Conference on Intelligent Robots and Systems, 2004.