**LANE DETECTION USING COMPUTER VISION FOR SELF DRIVING CARS**

**NAME**: CHITTIBOMMA BALARAM

**BRANCH:** COMPUTER SCIENCE OF ENGINEERING AND TECHNOLOGY

**YEARN**: 3RD YEAR

**ROLL NO**: 18R11A05A6

**COLLEGE**: GEETHANJALI COLLEGE OF ENGINEERING AND TECNOLOGY

**TECHNOLOGY USED:** PYTHON PROGRAMMING

**Skills required for the project:**

OPENCV,FLASK INTEGRATION,PYTHON PROGRAMMING

INDEX

1. **Introduction**
2. **Literature Survey** 
   1. **Project Literature**
   2. **Introduction To Python**
      1. **Python Technology**
      2. **Mvc Architecture**
      3. **Tkinter**
      4. **Libraries Specific To Project**
         1. **Imgutils**
         2. **Opencv**
         3. **Os**
3. **System Analysis And Requirements**
   1. **Feasibility Study**
      1. **Economical Feasibility**
      2. **Technical Feasibility**
      3. **Social Feasibility**
   2. **Software And Hardware Requirements**
      1. **Hardware Requirements**
      2. **Software Requirements**
   3. **Performance Requirements**
4. **Software Design**
   1. **Introduction**
   2. **Module Description**
5. **Coding Templates / Code**
   1. **App Code**
      1. main view
      2. load view
6. **Results And Validation**
   1. **Output Screens**
7. **Conclusion**

**Abstract:**

Driver support system is one of the most important feature of the modern vehicles to ensure driver safety and decrease vehicle accident on roads. Apparently, the road lane detection or road boundaries detection is the complex and most challenging tasks. It is includes the localization of the road and the determination of the relative position between vehicle and road. A vision system using on-board camera looking outwards from the windshield is presented in this paper. The system acquires the front view using a camera mounted on the vehicle and detects the lanes by  applying  few  processes. The lanes are extracted using Hough transform through a pair of hyperbolas which are   fitted to the edges of the lanes. The proposed lane detection system can be applied on both painted    and unpainted roads as well as curved and straight road in different weather conditions. The proposed system does not require any extra information such as lane width, time to lane crossing and offset between the center of the lanes. In addition, camera calibration and coordinate transformation are also not required. The system was investigated under various situations of changing illumination, and shadows effects in various road types without speed limits. The system has demonstrated a robust performance for detecting the road lanes under differentconditions.

1. **INTRODUCTION**

Advanced Driving Assistance Systems (ADAS) require the ability to model the shape of road lanes and localize the vehicle with respect to the road. Although, the main reason to build intelligent vehicles is to  improve the safety conditions by the entire or partial automation of driving tasks. Among these tasks, the road detection took an important role in driving assistance systems that provides information such as lane structure and vehicle position relative to the lane. However, vehicle crashes remains the leading cause of accident death and injuries in Malaysia and Asian countries which claiming tens of thousands of lives and injuring millions     of people each year. Most of these transportation deaths and injuries occur on the nation’s highways. The  United Nations has ranked Malaysia as 30th among countries with the highest number of fatal road accidents, registering an average of 4.5 deaths per 10,000 registered vehicles (Benozzi et al., 2002). Therefore, a system that provides a means of warning to a driver for a danger has been considered as a potential way to save a considerable number of lives. One of the main technology involves in these tasks is computer vision which becomes a powerful tool for sensing the environment and has been widely used in many applications by the intelligent transportation systems (ITS). In some proposed systems such as Tsugawa and Sadayuki, (1994), the lane detection consists of the localization of specific primitives such as the road markings of the surface of painted roads. Some systems achieves good results, but detecting the road lane remains a challenging task   under adverse conditions (heavy rain, degraded lane markings, adverse meteorological and lighting conditions) that are often met in real driving situations. Under such conditions, the system should at least switch off automatically and not report a false detection, nevertheless, two situations can disturb the process. The presence of other vehicles on the same lane may occlude partially the road markings ahead of the vehicle are the   presence of shadows caused by trees, buildings etc. This paper presents a vision- based approach which is capable of reaching a real time performance in detecting and tracking of structured  road boundaries (painted    or unpainted lane markings) with slight curvature and shadow conditions. Road boundaries are detected by fitting a parallel hyperbola pairs to the edges of the lane after applying the edge detection and  Hough  transform.The vehicle is supposed to move on a flat and straightroad or with slow curvature.

**Lane detection** is a critical component of self-driving cars and autonomous vehicles. Once lane positions are obtained, the vehicle will know where to go and avoid the risk of running into other lanes or getting off the road. This can prevent the driver/car system from drifting off the driving lane. There are multiple ways we can perform lane detection. We can use the learning-based approaches, such as training a deep learning model, However, there are simpler methods to perform lane detection as well. In this guided project, we will guide you in detecting lanes using Computer Vision with the popular OpenCV library in python.

**2.LITERATURE SURVEY**

**2.1 PROJECT LITERATURE**

     Literature survey is the most important step in software development process. Before developing the tool it is necessary to determine the time factor, economy and company strength. Once these things are satisfied, then next step is to determine which operating system and language can be used for developing the tool. Once the programmers start building the tool the programmers need lot of external support. This support can be obtained from senior programmers, from book or from websites. Before building the system the above consideration are taken into account for developing the proposed system.

**2.2. INTRODUCTION TO PYTHON**

Python is a popular programming language, Python is a high-level, interpreted, interactive and object oriented-scripting language. Python is Interpreted Python is Interactive Python is Object-Oriented Python is Beginner's Language Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands. Python's feature highlights include: Easy-to-learn Easy-to-read Easy-to-maintain A broad standard library Interactive Mode Portable Extendable Databases GUI Programming Scalable

Python language is a high level language that can be characterized by the following buzzwords.

* Object-Oriented
* Scripting
* Rapid Prototyping
* Steering
* Rapid Application Development
* Productivity

With most programming languages, you either compile or interpret a program so that you can run it on your computer.

* **Python is Interpreted** − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
* **Python is Interactive** − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
* **Python is Object-Oriented** − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
* **Python is a Beginner's Language** − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

Python enables programs to be written compactly and readably. Programs written in Python are typically much shorter than equivalent C, C++, or Java programs, for several reasons:

    • the high-level data types allow you to express complex operations in a single statement;

    • statement grouping is done by indentation instead of beginning and ending brackets;

     • no variable or argument declarations are necessary

1. **Python Features**

           Python's features include −

* **Easy-to-learn** − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
* **Easy-to-read** − Python code is more clearly defined and visible to the eyes.
* **Easy-to-maintain** − Python's source code is fairly easy-to-maintain.
* **A broad standard library** − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
* **Interactive Mode** − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
* **Portable** − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
* **Extendable** − You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
* **Databases** − Python provides interfaces to all major commercial databases.
* **GUI Programming** − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
* **Scalable** − Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below −

* It supports functional and structured programming methods as well as OOP.
* It can be used as a scripting language or can be compiled to byte-code for building large applications.
* It provides very high-level dynamic data types and supports dynamic type checking.
* It supports automatic garbage collection.
* It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

**2.2.1 Python Technology**

Python is everywhere. You may not even realize how widespread it is. Most of all, [Python is easy to learn, clear to read, and simple to write in](https://stxnext.com/ebooks/introduction-python-tech-managers/). This speeds up development without sacrificing reliability or scalability. It can be used in the following :

1. Artificial Intelligence with Python
2. Bio python
3. Bokeh
4. Beautiful Soup
5. Concurrency in python
6. Cryptography
7. Jupyter
8. Jython
9. Logistic Regression in Python
10. Matplotlib
11. Machine Learning with Python
12. NumPy
13. Object Oriented Python
14. PyGTK
15. PyQt
16. Pycharm
17. Python MYSQL

List of Python Web Frameworks:

**1. Full-Stack Frameworks**

This type of framework acts as a one-stop solution for fulfilling all the developers’ necessary requirements. Form validation, form generators, and template layouts are the components that are commonly included in the full-stack frameworks.

* CubicWeb
* [Django](https://mindmajix.com/python-django-training)
* Giotto
* Pylons Framework
* Pyramid
* TurboGears
* Web2Py

**2. Microframeworks**

These frameworks are also called non-full stack frameworks. These are lightweight in nature because they do not offer more components as a full-stack framework. The developers who want to use this framework must put a lot of effort in coding and also in adding additional requirements manually.

* Bottle
* CherryPy
* Dash
* Falcon
* Flask
* Hug
* MorePath
* Pycnic

**3. Asynchronous Framework**

It is also a microframework that mainly allows us in handling a huge set of concurrent connections. This asynchronous framework is mainly built for Python and it also uses the asyncio library.

* AIOHTTP
* Growler
* Sanic
* Tornado

**2.2.2 MVC Architecture**

MVC is a widely used software architectural pattern in GUI-based applications. It has three components, namely a ***model*** that deals with the business logic, a ***view*** for the user interface, and a ***controller*** to handle the user input, manipulate data, and update the view. The following is a simplified schematic that shows the basic interactions between the various components:

1. Model:

It consists of pure application logic, which interacts with the database. It includes all the information to represent data to the end user.

1. View:

View represents the HTML files, which interact with the end user. It represents the model’s data to user.

1. Controller:

It acts as an intermediary between view and model. It listens to the events triggered by view and queries model for the same.

**2.2.3 Tkinter**

Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit

1. Tkinter Widgets

Tkinter provides various controls, such as buttons, labels and text boxes used in a GUI application. These controls are commonly called widgets.

There are currently 15 types of widgets in Tkinter. We present these widgets as well as a brief description in the following table –

|  |  |
| --- | --- |
| 1 | [Button](https://www.tutorialspoint.com/python/tk_button.htm)  The Button widget is used to display buttons in your application. |
| 2 | [Canvas](https://www.tutorialspoint.com/python/tk_canvas.htm)  The Canvas widget is used to draw shapes, such as lines, ovals, polygons and rectangles, in your application. |
| 3 | [Checkbutton](https://www.tutorialspoint.com/python/tk_checkbutton.htm)  The Checkbutton widget is used to display a number of options as checkboxes. The user can select multiple options at a time. |
| 4 | [Entry](https://www.tutorialspoint.com/python/tk_entry.htm)  The Entry widget is used to display a single-line text field for accepting values from a user. |
| 5 | [Frame](https://www.tutorialspoint.com/python/tk_frame.htm)  The Frame widget is used as a container widget to organize other widgets. |
| 6 | [Label](https://www.tutorialspoint.com/python/tk_label.htm)  The Label widget is used to provide a single-line caption for other widgets. It can also contain images. |
| 7 | [Listbox](https://www.tutorialspoint.com/python/tk_listbox.htm)  The Listbox widget is used to provide a list of options to a user. |
| 8 | [Menubutton](https://www.tutorialspoint.com/python/tk_menubutton.htm)  The Menubutton widget is used to display menus in your application. |
| 9 | [Menu](https://www.tutorialspoint.com/python/tk_menu.htm)  The Menu widget is used to provide various commands to a user. These commands are contained inside Menubutton. |
| 10 | [Message](https://www.tutorialspoint.com/python/tk_message.htm)  The Message widget is used to display multiline text fields for accepting values from a user. |
| 11 | [Radiobutton](https://www.tutorialspoint.com/python/tk_radiobutton.htm)  The Radiobutton widget is used to display a number of options as radio buttons. The user can select only one option at a time. |
| 12 | [Scale](https://www.tutorialspoint.com/python/tk_scale.htm)  The Scale widget is used to provide a slider widget. |
| 13 | [Scrollbar](https://www.tutorialspoint.com/python/tk_scrollbar.htm)  The Scrollbar widget is used to add scrolling capability to various widgets, such as list boxes. |
| 14 | [Text](https://www.tutorialspoint.com/python/tk_text.htm)  The Text widget is used to display text in multiple lines. |
| 15 | [Toplevel](https://www.tutorialspoint.com/python/tk_toplevel.htm)  The Toplevel widget is used to provide a separate window container. |
| 16 | [Spinbox](https://www.tutorialspoint.com/python/tk_spinbox.htm)  The Spinbox widget is a variant of the standard Tkinter Entry widget, which can be used to select from a fixed number of values. |
| 17 | [PanedWindow](https://www.tutorialspoint.com/python/tk_panedwindow.htm)  A PanedWindow is a container widget that may contain any number of panes, arranged horizontally or vertically. |
| 18 | [LabelFrame](https://www.tutorialspoint.com/python/tk_labelframe.htm)  A labelframe is a simple container widget. Its primary purpose is to act as a spacer or container for complex window layouts. |
| 19 | [tkMessageBox](https://www.tutorialspoint.com/python/tk_messagebox.htm)  This module is used to display message boxes in your applications. |

1. **Standard attributes:**

* Dimensions
* Colors
* Fonts
* Anchors
* Relief styles
* Bitmaps
* Cursors

**2.2.4  Libraries Specific To Project**

**2.2.4.1 Imgutils**

Imutils are a series of convenience functions to make basic image processing functions such as translation, rotation, resizing, skeletonization, and displaying Matplotlib images easier with OpenCV and both Python 2.7 and Python 3

**2.2.4.2 Open CV**

OpenCV does not have any dedicated modules that can be used to read and decode barcodes and QR codes.OpenCV can do is facilitate the process of reading barcodes and QR codes, including loading an image from disk, grabbing a new frame from a video stream, and processing it.

**3.SYSTEM ANALYSIS AND REQUIREMENTS**

**3.1 FEASIBILITY STUDY:**

           The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis, the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company.  For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

1. ECONOMICAL FEASIBILITY
2. TECHNICAL FEASIBILITY
3. SOCIAL FEASIBILITY

**3.1.1 ECONOMICAL FEASIBILITY**

  This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

**3.1.2 TECHNICAL FEASIBILITY**

       This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

**3.1.3 SOCIAL FEASIBILITY**

             The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

* 1. **SOFTWARE AND HARDWARE REQUIREMENTS**

**3.2.1 Hardware Requirements:**

* Processor                   :  Core 2 Duo or Higher
* RAM                            :   1 GB
* Hard disk                    :   10 GB

**3.2.2 Software Requirements:**

* Operating Systems :  Any Graphical OS
* Language                  :  Python
* Database                   :  SQLite
* Tool                            : Pychram or Microsoft Visual Code

**3.3 PERFORMANCE REQUIREMENTS**

Performance is measured in terms of the output provided by the application. Requirement specification plays an important part in the analysis of a system. Only when the requirement specifications are properly given, it is possible to design a system, which will fit into required environment. It rests largely with the users of the existing system to give the requirement specifications because they are the people who finally use the system.  This is because the requirements have to be known during the initial stages so that the system can be designed according to those requirements.  It is very difficult to change the system once it has been designed and on the other hand designing a system, which does not cater to the requirements of the user, is of no use.

         The requirement specification for any system can be broadly stated as         given below:

The system should be able to interface with the existing system

1. The system should be accurate
2. The system should be better than the existing system

The existing system is completely dependent on the user to perform all the duties.

**4. SOFTWARE DESIGN**

**4.1 INTRODUCTION**

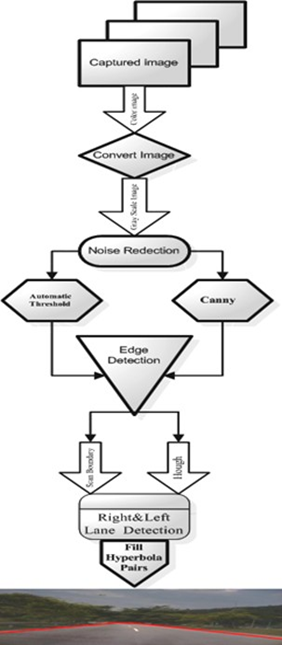
**software design** is the process or art of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. One could see it as the application of systems theory to product development. There is some overlap and synergy with the disciplines of systems analysis, systems architecture and systems engineering.

**4.3Module Description:**

A module is a separate unit of software or hardware. Typical characteristics of modular components include portability, which allows them to be used in a variety of systems, and interoperability, which allows them to function with the components of other systems. The term was first used in architecture.

Modular programming is the concept that similar functions should be contained within the same unit of programming code and that separate functions should be developed as separate units of code so that the code can easily be maintained and reused by different programs. Object-oriented programming is a newer idea that inherently encompasses modular programming.

***The Proposed Algorithm:***

The proposed algorithm structure is shown  in  Figure 2. A CCD camera is fixed on the front-view mirror  to capture the road scene. To simplify the problem, the baseline is assumed to be setup as horizontal, which assures the horizon in the image is parallel to the *x*-axis. Otherwise, the image of the camera can be adjusted using the calibration data. Each lane boundary marking, usually a rectangle (or approximate) forms a pair of edge lines.

**Fig. 2:** An over view of the proposed algorithm

In this paper, it was assumed that the input to the algorithm was a 620x480 RGB color image. Therefore  the algorithm works to convert the image to a grayscale image in order to minimize the processing time. Secondly, in presence of noise, the image will hinder the correct edge detection. Hence, F.H.D algorithm Mohamed Roushdy (2007) was applied to make the edge detection more accurate. Then the edge detector was used to produce an edge image by using canny filter with an automatic thresholding to obtain the edges. It      has reduced the amount of learning data required by simplifying the image edges considerably. Then edged image has been sent to the line detector which produces a right and left lane boundary segment. The projected intersection of these two line segments was determined and was referred to as the horizon. The lane boundary scan used the information in the edge image detected by the Hough transform to perform the scan. The scan returned a series of points on the right and left side. Finally pair of hyperbolas were fitted to these data points   to represent the lane boundaries. For visualization purposes the hyperbolas are displayed on the original color image.

***Image Capturing:***

The input data was a color image sequences taken from a moving vehicle. A color camera was mounted inside the vehicle at the front-view mirror along the central line. It took 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 captured the images in real time (up to 30 frames/second), and saved them in the computer memory. The lane detection system read the image sequences  from the  memory and started processing. A typical sceneof the road ahead is depicted by Figure 1.

***Conversion to GrayScale:***

To retain the color information as well as to segment the road from the lane boundaries using the color information, edge detection becomes difficult and consequently effects the processing time. In practice the road surface  can be made up of many different colors due to shadows, 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 were converted into grayscale. However, the processing of grayscale images became minimal as compared to a color image. This function transformed a 24-bit, three-channel, color image to an 8-bit, single channel grayscaleimage. The functionformed a weightedsum of the Red componentof the pixel value \*

0.3 +Green component of the pixel value \* 0.59 + Blue component for the pixel value \*0.11 and the output       is the gray scalevalue for the corresponding pixel(www.themalaysian.blogspot.com).

***Noise Reduction:***

Noise is a real world problem for all systems including computer vision processing. The developed algorithms must either be noise tolerant or the noise must be eliminated. As presence of noise in proposed system will hinder the correct edge detection. Hence noise removal is a pre requisite for  efficient  edge detection with the help of (F.H.D.) algorithm Mohamed Roushdy, (2007) that removed strong shadows from      a single image. The basic idea was that a shadow has a distinguished boundary. Hence removing the shadow boundary from the image derivatives and reconstructing the image was applied. A shadow edge image has been created by applying edge-detection on the invariant image and the original image. By selecting the edges that exist in the original image but not in the invariant image and to reconstruct the shadow  free  image  by removingthe edges from the originalimage by using a pseudo-inverse filter has been implemented.

***Edge Detection:***

Lane boundaries are defined by sharp contrast between the road surface and painted lines or some types     of non-pavement surfaces. These sharp contrasts are edges in the images. 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 was implemented for this algorithm. The one that produced the best edge images from all  the  evaluatededge detectors was the ‘canny’edge detector.

It was important to have the edge detection algorithm that could be able to select thresholds automatically However, the automatic threshold used in the default Canny Algorithm produced edge information that is far from actual threshold. A slight modification to the edge detection in canny has produced more desirable results. The only changes  necessary were to set the amount of non-edge pixels of the highest and lower thresholding    to the best value that has provided more accurate edges in different conditions of image capturing environment. The process is given in equation (1) and an image detected by  modified Canny Algorithm is presented in  Figure 3

**Fig. 3:** An image detected by modified Canny Algorithm

**5.** **code**

5.1 App code

i.Main view

from tkinter import \*

from PIL import Image,ImageTk

import os

import cv2

from tkinter import filedialog

from lane import videoLanes

def open\_image():

     x=openfilename()

result=videoLanes(x)

def openfilename():

     filename = filedialog.askopenfilename(title ='open')

     return filename

window=Tk()

window.title("load video")

window.config(bg='skyblue')

window.geometry('300x300')

l1=Label(window,text='Please upload the video by clicking the below button').pack()

b1=Button(window,text="Please upload the video",command=open\_image)

b1.pack(padx=20,pady=20)

window.mainloop()

ii.Lane.py

import cv2 as cv

import numpy as np

from matplotlib import pyplot as plt

def lanesDetection(img):

    height = img.shape[0]

    width = img.shape[1]

    region\_of\_interest\_vertices = [

        (200, height), (width/2, height/1.37), (width-300, height)

    ]

    gray\_img = cv.cvtColor(img, cv.COLOR\_RGB2GRAY)

    edge = cv.Canny(gray\_img, 50, 100, apertureSize=3)

    cropped\_image = region\_of\_interest(

        edge, np.array([region\_of\_interest\_vertices], np.int32))

    lines = cv.HoughLinesP(cropped\_image, rho=2, theta=np.pi/180,

                           threshold=50, lines=np.array([]), minLineLength=10, maxLineGap=30)

    image\_with\_lines = draw\_lines(img, lines)

    return image\_with\_lines

def region\_of\_interest(img, vertices):

    mask = np.zeros\_like(img)

    match\_mask\_color = (255)

    cv.fillPoly(mask, vertices, match\_mask\_color)

    masked\_image = cv.bitwise\_and(img, mask)

    return masked\_image

def draw\_lines(img, lines):

    img = np.copy(img)

    blank\_image = np.zeros((img.shape[0], img.shape[1], 3), np.uint8)

    for line in lines:

        for x1, y1, x2, y2 in line:

            cv.line(blank\_image, (x1, y1), (x2, y2), (0,250, 0), 2)

    img = cv.addWeighted(img, 0.8, blank\_image, 1, 0.0)

    return img

def videoLanes(video):

    cap = cv.VideoCapture(video)

    while(cap.isOpened()):

        ret, frame = cap.read()

        frame = lanesDetection(frame)

        cv.imshow('Lanes Detection', frame)

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

            break

    cap.release()

    cv.destroyAllWindows()

**7.RESULTS AND VALIDATION**

This section evaluates the overall performance of the system, after solving most of the problems discovered in earlier developed stages of the scheme. The performance of the algorithm is evaluated qualitatively in terms of accuracy in the localization of the lane boundaries for 150 frames in each case. This tier performance metric per input frame is strictly a pass/fail vote based  on the likelihood that a vehicle could conceivably navigate  with the output hyperbola pairs. It has been developed from the algorithm where accurate lane detection is marked by a red line at 2 and once it drops down to 1 it indicates the fault lane detection at that frame   sequence.

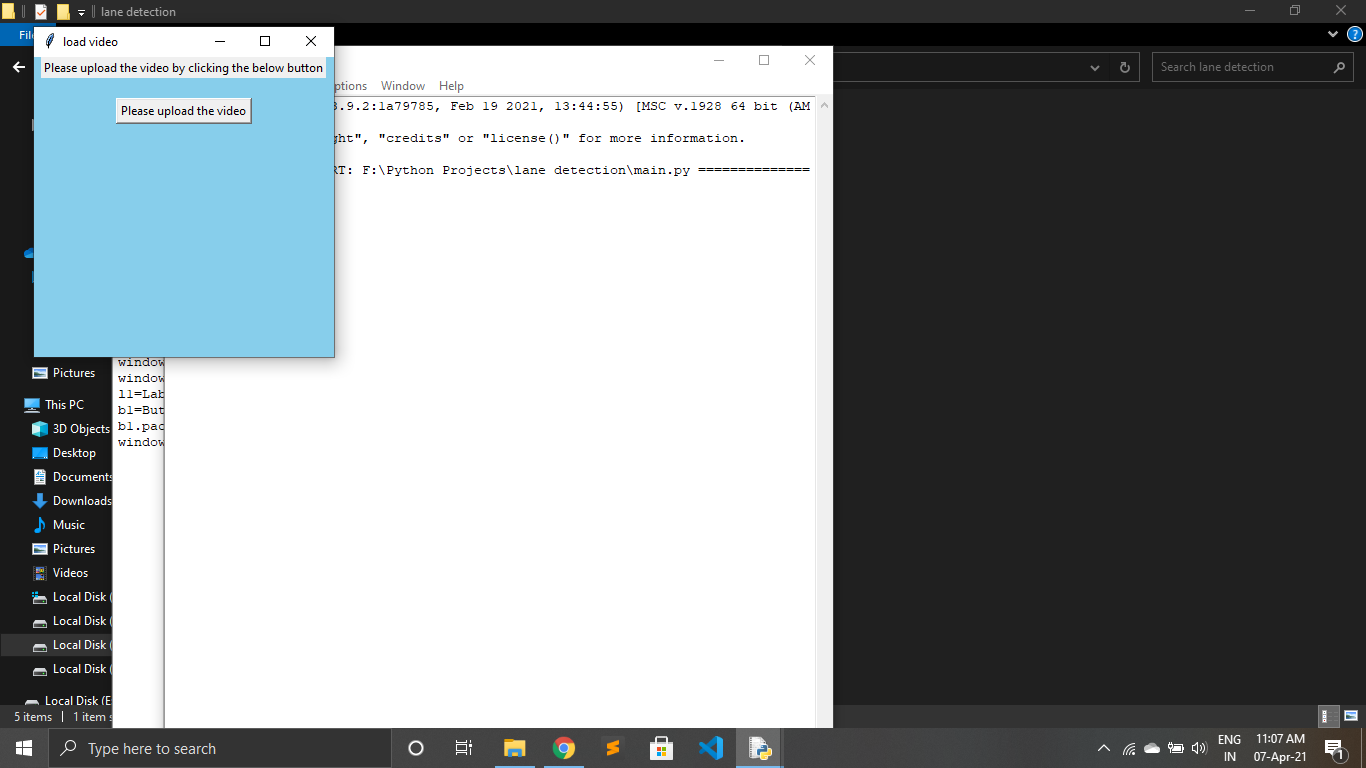
The final summary of the best results obtained in different road conditions in this project were shown in Figure 12 where the detection rate per frame achieved 96.6% during day time and 92.6% at night time as presented in Figures 13 and 14. The results is considered to be satisfactory after avoiding most of the light reflection problems that caused to have a very disappointing results at the preliminary stage.

A successful completion of the algorithm has been  shown in Figure 15 where the lanes were disturbed      by variety of noses ranging from shadows in the lane to vehicle on the road causing the lanes to be indistinguishable. Sharp curves caused some frames to fail during detection. The failures in the test frames  came from a multitude of sources including line extraction problems as well as problems in finding the lane boundary pixels. However, the lane detection rate was still in stable condition obtaining above 95% with different noise cases showingthe strength of the algorithm in handling noise as depictin Figure 16.

There are satisfactory results for critical weather conditions such as cloudy or rainy, as presented in Figure

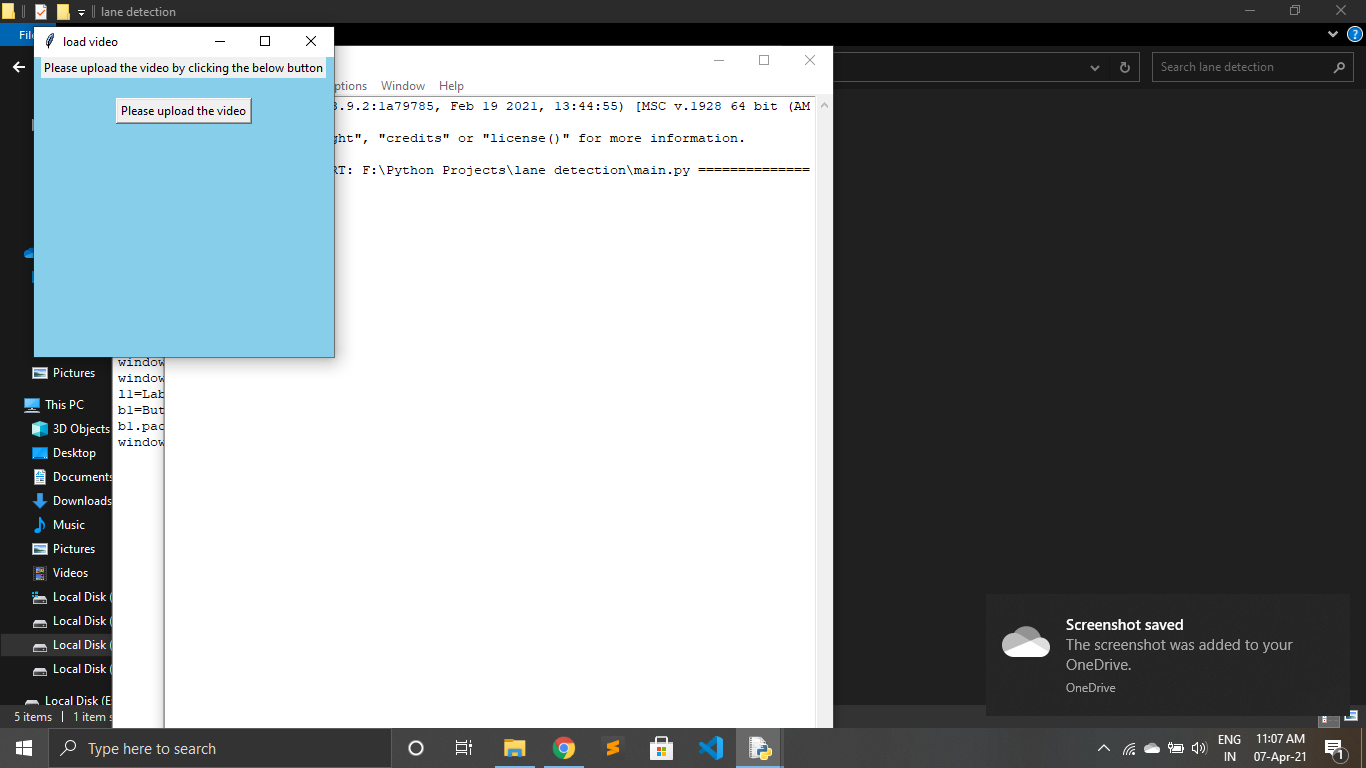
17 where the lanes were detected in cloudy weather and during heavy rain. The lanes were scarcely distinguished in the road. The most elusive problem causing low detection is the moving wiper of the windshield. However, the system is accurately detecting the lanes after the wiper moves up or down, meaning   it still produces acceptable results under the poor visibility condition approaching 75% per frame. Figure 18 illustrates the above mentioneddetection rate.

a. **OUTPUT SCREENS**

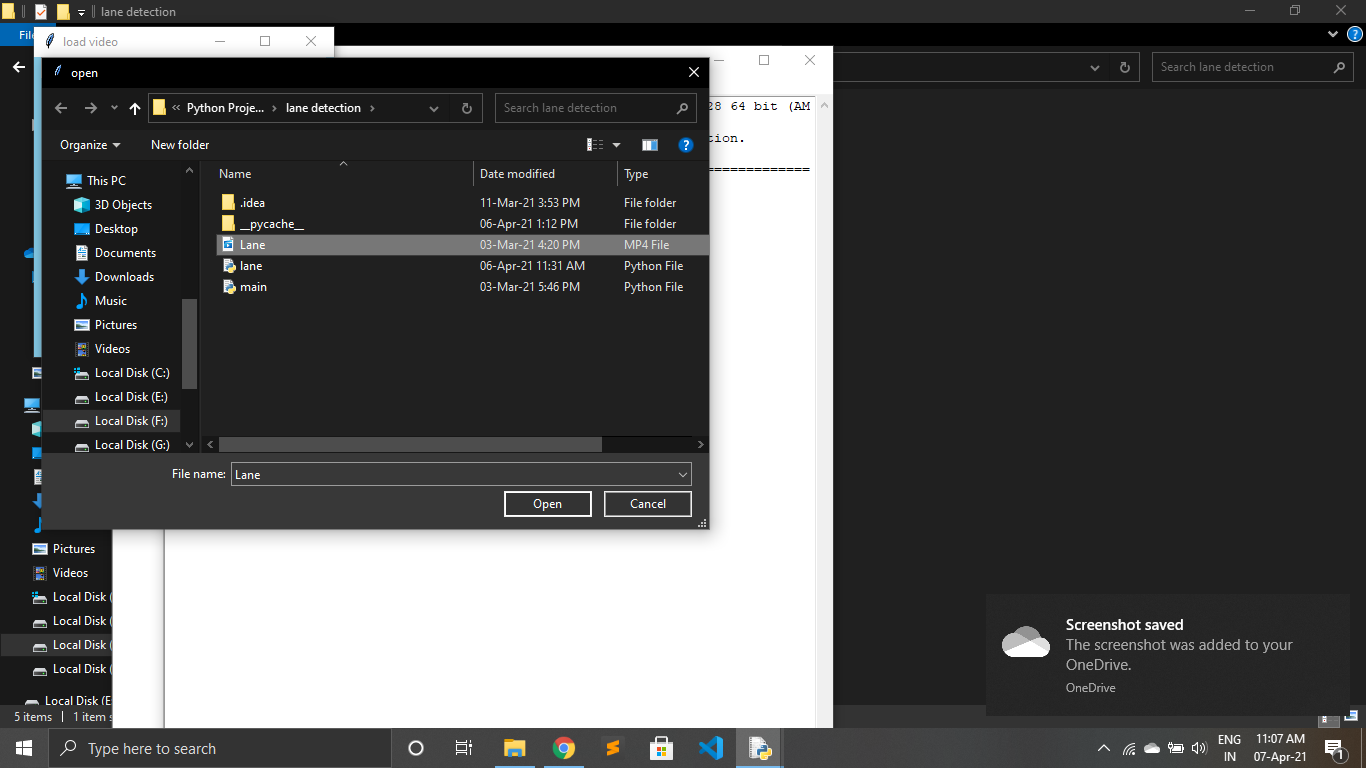
 **EXECUTING THE APPLICATION**

**-EXECUTE THE APPLICATION IN ANY IDE**

**-RUN THE APPLICATION BY USING THE COMMAND MAIN.PY**

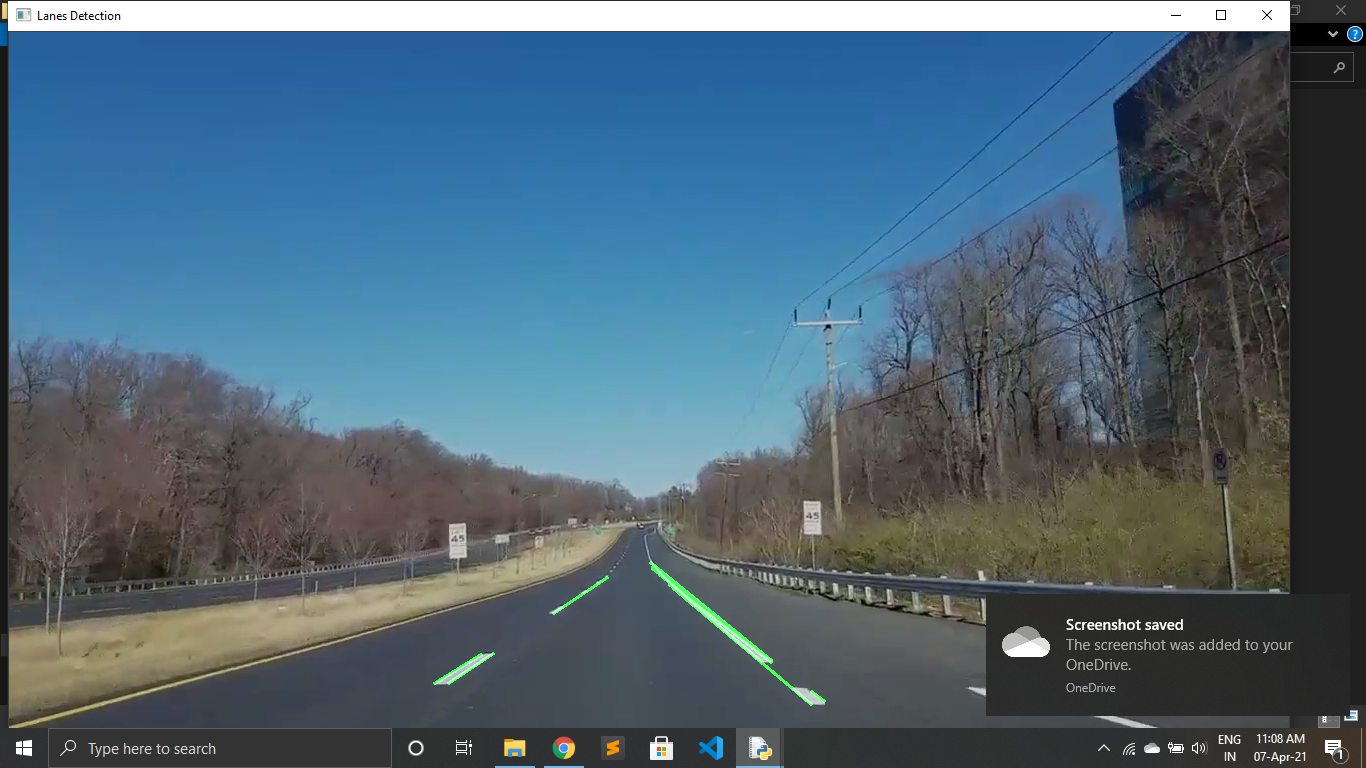
 **UPLOAD THE VIDEO BY CLICKING THE BUTTON**

**-UPLOAD THE VIDEO BY CLICKING ON PLEASE UPLOAD THE VIDEO BUTTON**

 **CHOOSE THE VIDEO FILE FROM ANY DIRECTORY**

**-A WINDOW WILL BE OPENED AFTER PRESSING ON UPLOAD BUTTON**

**-CHOOSE THE VIDEO FROM ANY DIRECTORY**

 **THE LANES WILL BE DETECTED IN THE RESULT VIDEO**

**-THE LANES WILL BE DETECTED IN THE UPLOAD VIDEO**

**Conlusion:**

In this paper, A real time vision-based lane detection method was proposed. Image segmentation and remove the shadow of the road were processed. Canny operator was used to detect edges that represent road lanes or road boundaries. A hyperbola-pair road model used to deal with the occlusion and imperfect road condition. A series of experiment showed that the lanes were detected using Hough transformation with restricted search area and the projection of their intersection will form the last scan point called the horizon. Furthermore, In order to search out for the left and right vector points that represent the road lanes, the lane   scan boundary phase uses the edge image and  the left and right   Hough lines and the horizon line as inputs,     to effectively allocate the lane points. That was demonstrated by two hyperbola lines. The experimental results showed that the system is able to achieve a standard requirement to provide valuable information to the driver  to ensure safety.