

Computer Vision/Graphics

A project report submitted in partial fulfilment of the requirements

for the award of credits to

Data-processing and Visualization using Python

a skill-oriented course of **Bachelor of Technology**

In

ELECTRONICS & COMMUNICATION ENGINEERING

By

PANNURL.GANESH VARDHAN
22BQ1A0494



DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY

(Approved by AICTE and permanently affiliated to JNTUK)

Accredited by NBA and NAAC with 'A' Grade

NAMBUR (V), PEDAKAKANI (M), GUNTUR-522 508

NOVEMBER 2023

**DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING
VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY: NAMBUR
JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY KAKINADA**



CERTIFICATE

This is to certify that the project titled “**COMPUTER VISION/GRAPHICS**” is a bonafide record of work done by **PANNURI.GANESHVARDHAN** under the guidance of **Sk.Mastanbi, Assistant Professor**, in partial fulfilment of the requirement for the award of credits to **Data Preprocessing and Visualization using Python-** a skill-oriented course of Bachelor of Technology in Electronics and Communication Engineering, JNTUK during the academic year 2023–24.

Sk.Mastanbi
Course Instructor

Dr. M.Y. Bhanu Murthy
Head of the Department

DECLARATION

I am **PANNURI.GANESH VARDHAN** hereby declare that the project report entitled “**COMPUTER VISION/GRAPHICS**” done by me under the guidance of **Sk.Mastanbi, Assistant Professor, Department of Electronics and Communication Engineering** is submitted by partial fulfilment of requirements for the award of degree of **BACHELOR OF TECHNOLOGY** in **ELECTRONICS AND COMMUNICATION ENGINEERING**.

Date: -

Place: - VVIT, Nambur

P.G.Vardhan

Signature of the candidate

ACKNOWLEDGEMENT

We express our sincere thanks wherever it is due

We express our sincere thanks to the Chairman, Vasireddy Venkatadri Institute of Technology, **Sri Vasireddy Vidya Sagar** for providing us well equipped infrastructure and environment.

We thank **Dr. Y. Mallikarjuna Reddy**, Principal, Vasireddy Venkatadri Institute of Technology, Nambur, for providing us the resources for carrying out the project.

We express our sincere thanks to **Dr. K. Giribabu**, Dean of Studies for providing support and stimulating environment for developing the project.

Our sincere thanks to **Dr. M. Y. Bhanu Murthy**, Head of the Department, Department of ECE, for his co-operation and guidance which help us to make our project successful and complete in all aspects.

We also express our sincere thanks and are grateful to our guide **Sk.Mastanbi**, Asst.Professors,Department of ECE, for motivating us to make our project successful and fullycomplete. We are grateful for his precious guidance and suggestions.

We also place our floral gratitude to all other teaching staff and lab technicians for their constant support and advice throughout the project.

NAME OF THE CANDIDATE:PANNURI.GANESH

VARDHAN

REGD NO: 22BQ1A0494

CONTENTS

LIST OF CONTENTS	PAGE NO.
LIST OF FIGURES	ii
ABSTRACT	iii
1.INTRODUCTION	1-3
2.LIBRARIES DESCRIPTION	4-6
2.1 numpy	
2.1.1 vital features of numpy	
2.2 opencv	
2.2.1 vital features of opencv	
3.IMPLEMENTATION	7-17
3.1 Main steps	
3.2 Loading dependices	
3.2 .1 Command prompt	
3.2.2 Upgrading pip	
3.2.3 Implementation of Opencv	
3.2.4 Implementation of Numpy	
3.3 steps to download python idle	
3.4 steps to download python idle	
3.5 Workflow	
3.6 Canny edge detection	
3.7 Hough tramform	
3.8 Loading data	
3.9 Data preprocessing	
4.RESULTS AND CONCLUSION	18-19
4.1 Results	
4.2 Conclusion	
REFERENCE	20-21
APPENDIX	22-24

LIST OF FIGURES

FIG NO.	TITLE	PAGENO:
Fig 1.1	RaspBerryPi Touchscreen	3
Fig3.2	Importing libraries	7
Fig3.2 .1	Command prompt	7
Fig 3.2.2	Upgrading pip	8
Fig 3.2.3	Implementation of Opencv	8
Fig 3.2.4	Implementation of Numpy	9
Fig 3.3.1	Downloading Python IDLE	10
Fig 3.3.2	Installing Python on your Windows	10
Fig 3.3.3	continue installation	11
Fig 3.3.4	Installation will start	11
Fig 3.3.5	Installation gets completed	11
Fig 3.4.1	Python interpreter window	12
Fig 3.4.2	Idle shell	12
Fig 3.4.3	Restart the shell	13
Fig3.4.4	Displaying error	14
Fig 3.5	Workflow	14
Fig 3.8	Loading data	17
Fig4.1	Detected Lines	18

ABSTRACT

Lane detection is a challenging problem. Many technical improvements have recently been made in the field of road safety, as accidents have been increasing at an alarming rate, and one of the major causes of such accidents is a driver's lack of attention. To lower the incidence of accidents and keep safe, technological breakthroughs should be made. One method is to use Lane Detection Systems, which function by recognizing lane borders on the road and alerting the driver if he switches to an incorrect lane marking. A lane detection system is an important part of many technologically advanced transportation systems.

Although it is a difficult goal to fulfil because to the varying road conditions that a person encounters, particularly while driving at night or in daytime. A camera positioned on the front of the car catches the view of the road and detects lane boundaries. The method utilized in this research the video, which is then used to recognize the lanes on the road. Several methods for detecting lane markings on the road have been presented.

CHAPTER 1

INTRODUCTION

Using the computer vision algorithms and python it will predict the width of the road .It will also tell the distance from the roadside and also guide for overtaking the vehicles. The trained machine become the powerful feature in the driverless car as the vehicle get proper assistance in the overtaking and parking also .It can also be used in the cruise control of the car make the more impact of the car and easy in controlling the car. Active safety is currently a key topic in the automotive industry, which fosters the development of Autonomous Vehicle functions.

Various advanced driver assistance systems (ADAS) and active safety systems have the potential to improve road safety, driver comfort, fuel economy, and traffic flow by assisting the driver during different driving conditions. It is estimated that human error is a contributing factor in more than 90% of all accidents. In order to save the human lives caused by road accidents, it is hence of interest to develop such systems using modeling and simulation 7tools which is quick and more efficient as compared to the real driving testing. Overtaking is one of the most complex maneuvers with the high risk of collision (75% human error) so the automation of this maneuver still remains one of the toughest challenges in the development of autonomous vehicles. Since overtaking is one of the complex maneuvers and so many factors affect it, the automation of this maneuver has been considered to be one of the toughest challenges in the development of autonomous vehicles. Overtaking involves a great interaction between both longitudinal (throttle and brake) and lateral (steering) actuators. Nowadays, in the field of driver assistance systems and automated driving, development approaches for lateral maneuver control are the very big challenge.

7 Computerized reasoning in autos has highlighted in numerous exploration tasks and trials have been led since the 1980s when first models for self-ruling autos were displayed via Carnegie Mellon College's Navlab. From that point forward,

there have been a lot of innovative headways in the field of self-governing vehicles and Navlab 11, the most recent auto by CMU's Route Research facility, is a 2000 Jeep Wrangler introduced with several pieces of equipment such as GPS, magnetometers, proximity laser sensors, and omni-directional camera . Stanley, an autonomous car created by Stanford University in cooperation with Volkswagen won first driver-less car racing challenge known as 'DARPA Grand Challenge' in 2005. Numerous studies have been led about vehicle robotization and Driver Assistance Systems.

A portion of the highlights of a self-governing vehicle incorporates Automatic Cruise Control, Automatic Parking, Collision Avoidance as well as Lane Departure Warning systems. Aside from the said, the potential advantages of autonomous cars incorporate decreased framework costs, expanded security, expanded consumer satisfaction, and a critical lessening in car accidents. . Some in favor of autonomous vehicles additionally trust that conveying robotics to the car will kill more typical wrongdoings like insurance scams and vehicle burglary.

On account of above mentioned benefits, many countries have taken a step forward to bring self-driving cars on the roads for public. UK, in 2014, announced that driver less cars² will be allowed on public roads and soon a prototype called 'Lutz pod' was launched as UK's first driver-less car [6] [7]. Moreover, In 2017, first autonomous vehicle was demonstrated at Christchurch Airport [8]. However, despite significant amount of benefits, there exist some foreseeable challenges in completely accepting self-driven vehicles. A few opposer believe that widespread adoption of driver-less cars will bring dearth of driving related jobs and will also compromise with the passenger's safety . It is also believed that an autonomous car will likely lead to the loss of privacy and increased risks of hacking attacks and terrorism.

It is understood that the discrepancy between people's beliefs of the necessary government intervention may cause a delay in accepting autonomous cars on the road . This will keep the drivers in control of the 8 car for some years to come.

Therefore, it is important to seek solutions to make their driving experience safer and easier. This thesis presents one such solution - Real time road lane lines detection in different weather conditions. A real time vision-based lane detection system can be used to assist the driver in locating the lanes or warning the driver if the vehicle goes out of lane. In this thesis, a Lane Detection algorithm is presented for real-time detection of road lane lines in various climate conditions. The algorithm runs on a 7 inch display attached to a RaspberryPi 3.0 computer with a Pi camera installed. The entire system is placed on the car's dashboard.

The Pi Camera is placed inside the car fixed to the windshield to capture the real time video while driving. The display screen and the RaspberryPi computer is powered by a 12V-to-5V car power supply port. Lane Detection algorithm presented in this thesis implements the concepts of Computer Vision - like gradient change and probabilistic Hough transform, to detect the lane lines in the road videos. Moreover, certain filtering techniques, such as filtering based on slope and intercept values, are used to determine the exact location of road lane lines. The algorithm is implemented in Python 2.7 with OpenCV 3.0.

The hardware used for this research is shown in the Fig. 1. A wooden board is used to hold the touchscreen display and the Raspberry Pi computer together. This wooden board is placed on the vehicle's dashboard. A Pi camera, attached to the Raspberry Pi.



Fig1.1. RaspBerryPi Touchscreen

CHAPTER 2

LIBRARIES DESCRIPTION

2.1 Numpy

NumPy, which stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of routines for processing those arrays. Using NumPy, mathematical and logical operations on arrays can be performed. This tutorial explains the basics of NumPy such as its architecture and environment. It also discusses the various array functions, types of indexing, etc. An introduction to Matplotlib is also provided. All this is explained with the help of examples for better understanding.

2.1.1 VITAL FEATURES OF NUMPY

This is High-performance N-dimensional array object

The most important feature of the NumPy library. It is the homogeneous array object. We perform all the operations on the array elements. The arrays in NumPy can be one dimensional or multidimensional.

- It contains tools for integrating code from C/C++ and Fortran

We can use the functions in NumPy to work with code written in other languages. We can hence integrate the functionalities available in various programming languages. This helps implement inter-platform functions.

- It contains a multidimensional container for generic data

Here generic data refers to the parameterized data type of arrays. It can perform functions on the generic data types. The arrays in NumPy are of homogenous nature. These array elements are assigned parameters. The parameters help increase the diversity of the arrays.

- It had data type definition capability to work with varied databases

We can work with arrays of different data types. We can use the dtype function to determine the data type and hence get a clear idea about the available data set.

2.2 Opencv

openCV-Python is a library of Python bindings designed to solve computer vision problems.

Python is a general purpose programming language started by **Guido van Rossum** that became very popular very quickly, mainly because of its simplicity and code readability. It enables the programmer to express ideas in fewer lines of code without reducing readability.

Compared to languages like C/C++, Python is slower. That said, Python can be easily extended with C/C++, which allows us to write computationally intensive code in C/C++ and create Python wrappers that can be used as Python modules. This gives us two advantages: first, the code is as fast as the original C/C++ code (since it is the actual C++ code working in background) and second, it is easier to code in Python than C/C++. OpenCV-Python is a Python wrapper for the original OpenCV C++ implementation.

OpenCV-Python makes use of **Numpy**, which is a highly optimized library for numerical operations with a MATLAB-style syntax. All the OpenCV array structures are converted to and from Numpy arrays. This also makes it easier to integrate with other libraries that use Numpy such as SciPy and Matplotlib.

2.2.1 IMPORTANT FEATURES OF OPENCV

- Capture and save videos.
- Process images (filter, transform)
- Perform feature detection.
- Detect specific objects such as faces, eyes, cars, in the videos or images.
- Analyze the video, i.e., estimate the motion in it, subtract the background, and track objects in it.
- Open-source
- Fast speed
- Easy to integrate
- Ease of coding
- Fast prototyping

CHAPTER 3

IMPLEMENTATION

3.1 MAIN STEPS INVOLVED IN BUILDING AUTOMATIC ROAD LANE LINE DETECTION USING OPENCV AND NUMPY

- Identify the business problem.
- Define the success criteria.
- Determine the appropriate computer vision techniques.
- Collect and label training and test images.
- Train and evaluate model.
- Deploy and test.
- Iterate on the solution.

First of all, we have to import the required libraries

3.2 LOADING DEPENDENCIES

As I am using PythonIdle for my project there is no need to install any libraries, we just need to import them.

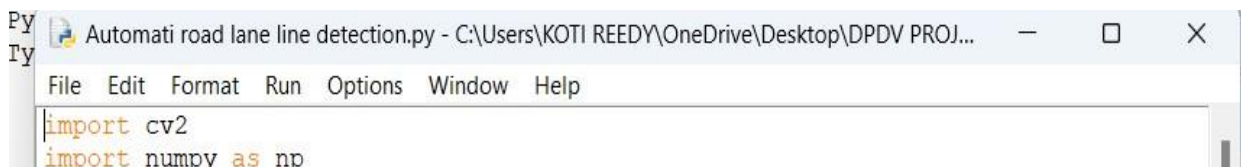
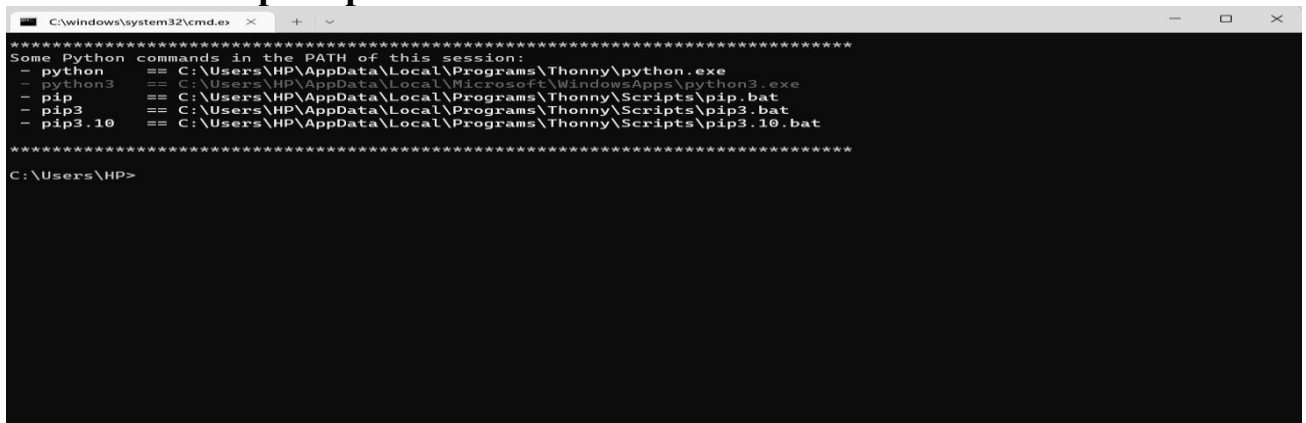


Fig 3.2 : Importing libraries

3.2.1 Command prompt

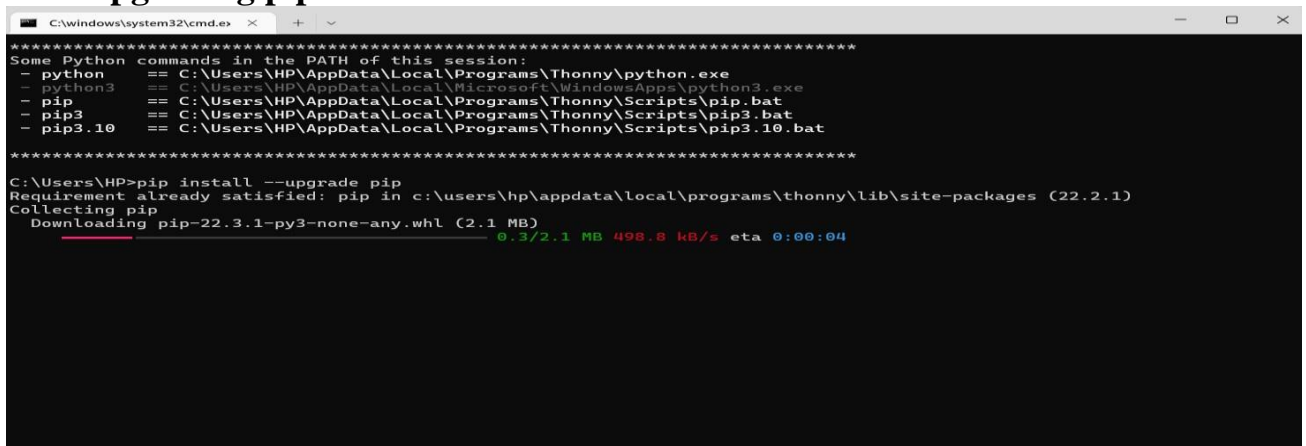


```
C:\windows\system32\cmd.exe
*****
Some Python commands in the PATH of this session:
- python == C:\Users\HP\AppData\Local\Programs\Thonny\python.exe
- python3 == C:\Users\HP\AppData\Local\Microsoft\WindowsApps\python3.exe
- pip == C:\Users\HP\AppData\Local\Programs\Thonny\Scripts\pip.bat
- pip3 == C:\Users\HP\AppData\Local\Programs\Thonny\Scripts\pip3.bat
- pip3.10 == C:\Users\HP\AppData\Local\Programs\Thonny\Scripts\pip3.10.bat
*****
C:\Users\HP>
```

Fig 3.2.1: Opening Command Prompt

The above figure shows that opening the command prompt in the windows.

3.2.2 Upgrading pip



```
C:\windows\system32\cmd.exe
*****
Some Python commands in the PATH of this session:
- python == C:\Users\HP\AppData\Local\Programs\Thonny\python.exe
- python3 == C:\Users\HP\AppData\Local\Microsoft\WindowsApps\python3.exe
- pip == C:\Users\HP\AppData\Local\Programs\Thonny\Scripts\pip.bat
- pip3 == C:\Users\HP\AppData\Local\Programs\Thonny\Scripts\pip3.bat
- pip3.10 == C:\Users\HP\AppData\Local\Programs\Thonny\Scripts\pip3.10.bat
*****
C:\Users\HP>pip install --upgrade pip
Requirement already satisfied: pip in c:\users\hp\appdata\local\programs\thonny\lib\site-packages (22.2.1)
Collecting pip
  Downloading pip-22.3.1-py3-none-any.whl (2.1 MB)
    0.3/2.1 MB 498.8 kB/s eta 0:00:04
```

Fig 3.2.3: Opening Command Prompt

In this figure the pip is getting updated.results the latest version to install very clearly

3.2.4. Implementation of Opencv

At first we should download Opencv.To download Opencv we use the Syntax as

“pip install –user opencv-python” or for latest version “python.exe -m pip install -- upgrade pip”.

```
C:\windows\system32\cmd.exe
*****
Some Python commands in the PATH of this session:
- python == C:\Users\HP\AppData\Local\Programs\Thonny\python.exe
- python3 == C:\Users\HP\AppData\Local\Programs\Thonny\Scripts\python3.exe
- pip == C:\Users\HP\AppData\Local\Programs\Thonny\Scripts\pip.exe
- pip3 == C:\Users\HP\AppData\Local\Programs\Thonny\Scripts\pip3.exe
- pip3.10 == C:\Users\HP\AppData\Local\Programs\Thonny\Scripts\pip3.10.exe
*****
C:\Users\HP\Downloads\DPVP PROJECT 2-1>pip install --upgrade pip
Requirement already satisfied: pip in c:\users\hp\appdata\local\programs\thonny\lib\site-packages (22.3.1)
C:\Users\HP\Downloads\DPVP PROJECT 2-1>pip install --user opencv-python
Collecting opencv-python
  Downloading opencv_python-4.6.0-66-cp36-abi3-win_amd64.whl (35.6 MB)
    26.6/35.6 MB 415.5 KB/s eta 0:00:22
```

Fig3.2.4 Implementation of Opencv

The above figure shows that how to install Opencv module.and the syntax is mentioned above the image.

3.2.5 Implementation of Numpy

At first we should download Numpy.To download Numpy we use the Syntax as

“pip install –numpy-python”

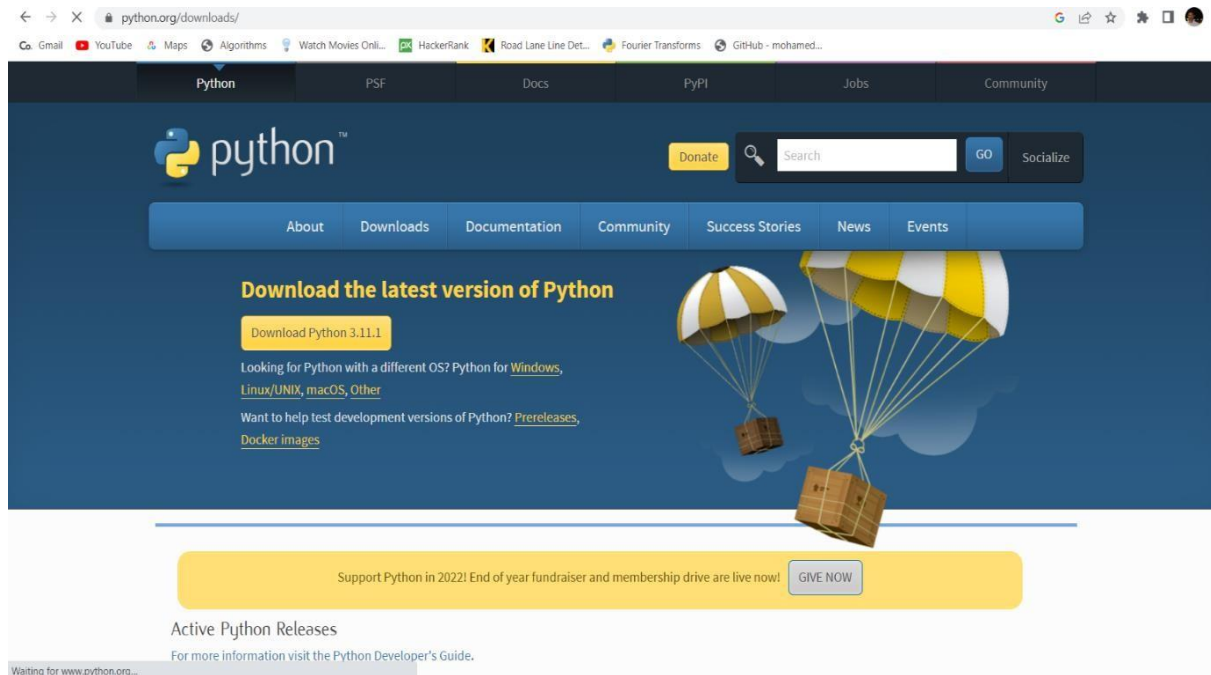
```
Command Prompt
Microsoft Windows [Version 10.0.22621.900]
(c) Microsoft Corporation. All rights reserved.

C:\Users\KOTI REEDY>pip install numpy
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: numpy in c:\users\koti reedy\appdata\roaming\python\python311\site-packages (1.23.4)
C:\Users\KOTI REEDY>
```

Fig3.2.5 Implementation of Numpy

3.3 STEPS TO DOWNLOAD PYTHON IDLE

1. Type Python.org or Python idle in chrome or any web browser and click on its website.



.Fig 3.3.1 Downloading

Installing Python on your Windows based computer.

- Double click at Python installer downloaded on your computer. Dialog box will appear as follow:

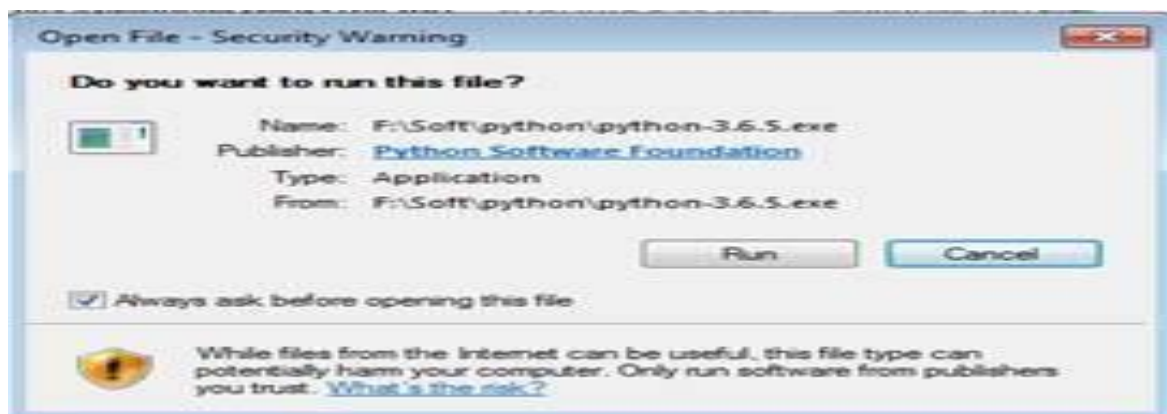


Fig 3.32 Dialog box

- Click at Run button to continue installation. Dialog Box will appear as



Fig 3.3.2 Asking to install

- Click at Install Now.
- PYthon installation will start and following window will appear



Fig 3.3.3 Installing

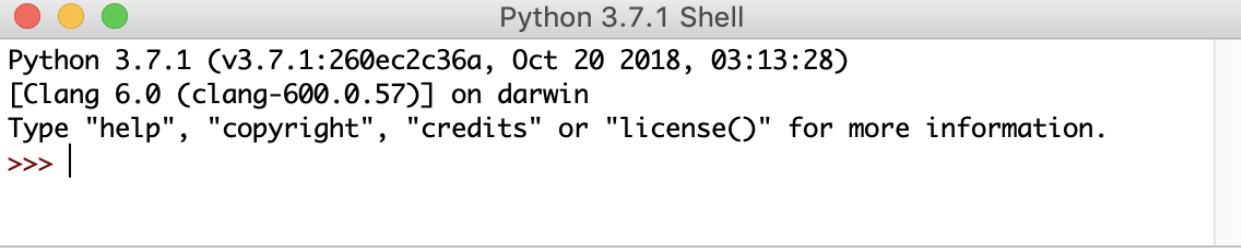
- When installation gets completed following window will appear



Fig 3.3.4 installation gets completed

3.4.How to Use the Python IDLE Shell

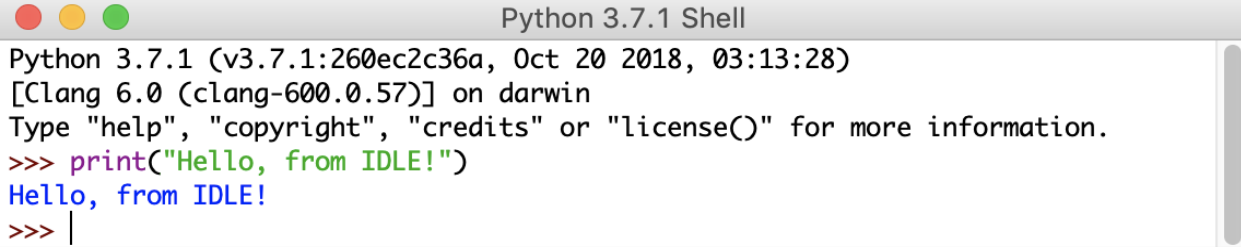
The shell is the default mode of operation for Python IDLE. When you click on the icon to open the program, the shell is the first thing that you see:



```
Python 3.7.1 (v3.7.1:260ec2c36a, Oct 20 2018, 03:13:28)
[Clang 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license()" for more information.
>>> |
```

Ln: 4 Col: 4

This is a blank Python interpreter window. You can use it to start interacting with Python immediately. You can test it out with a short line of code:



```
Python 3.7.1 (v3.7.1:260ec2c36a, Oct 20 2018, 03:13:28)
[Clang 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license()" for more information.
>>> print("Hello, from IDLE!")
Hello, from IDLE!
>>> |
```

Ln: 6 Col: 4

Fig3.4.1 Python interpreter window

Here, you used `print()` to output the string "Hello, from IDLE!" to your screen. This is the most basic way to interact with Python IDLE. You type in commands one at a time and Python responds with the result of each command.

Next, take a look at the menu bar. You'll see a few options for using the shell:

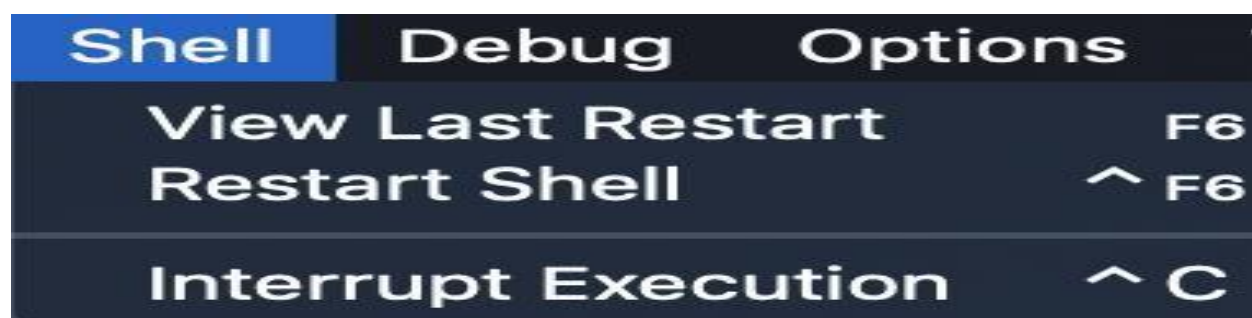
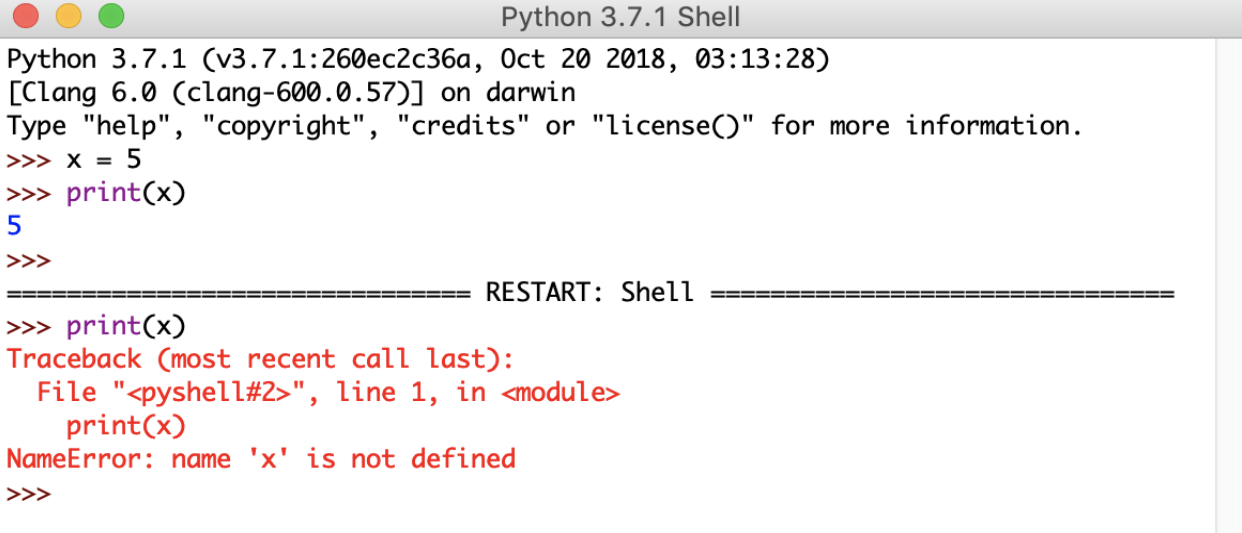


Fig3.4.2 Idle shell

You can restart the shell from this menu. If you select that option, then you'll clear the state of the shell. It will act as though you've started a fresh instance of Python IDLE. The shell will forget about everything from its previous state:



```
Python 3.7.1 Shell
Python 3.7.1 (v3.7.1:260ec2c36a, Oct 20 2018, 03:13:28)
[Clang 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license()" for more information.
>>> x = 5
>>> print(x)
5
>>>
===== RESTART: Shell =====
>>> print(x)
Traceback (most recent call last):
  File "<pyshell#2>", line 1, in <module>
    print(x)
NameError: name 'x' is not defined
>>>
```

Ln: 14 Col: 4

Fig3.4.3 restart the shell

In Fig3.4.3. restart the shell, you first declare a [variable](#), `x = 5`. When you call `print(x)`, the shell shows the correct output, which is the number 5. However, when you restart the shell and try to call `print(x)` again, you can see that the shell prints a [traceback](#). This is an error message that says the variable `x` is not defined. The shell has forgotten about everything that came before it was restarted. You can also interrupt the execution of the shell from this menu. This will stop any program or statement that's running in the shell at the time of interruption. Take a look at what happens when you send a keyboard interrupt to the shell:

3.6 CANNY EDGE DETECTION

This process consists of mainly four functions in it. The image can be of any type which consists of change in intensity in it. The change in intensity of pixels in an image defines the edges in an image. The canny edge detection mainly focuses on change in intensity in an image. The change in pixel's intensities from high intensity to low intensity is known as edge. At first, the color image is changed into black and white image and is passed to smoothening technique. We use Gaussian blur as a smoothening technique followed by gradient calculations, Nonmaximum suppression and double threshold. Edge detection mainly use derivatives of pixel intensities in an image and then reduce the complexity of an image. The edge is detected when there is change in intensity from high to low which refers white shades to black shades (in gray scale image) in an image. Gray Scale image is used because it would be easy to process the gray scale image than the colored image. A gradient calculation process is used for calculating the θ through Sobel filters. Non-Maximum suppression is a process of thinning of edges that should be occurred in a required output image. Then a double thresholding is done to intensify the strong pixels of an output image and to close the intensity of weaker pixels in an image. Thus, canny edge detection is used and its architecture is built.

3.7:Hough Transformations

Hough Transformations require a Hough transformation space which is used to rotate the angles of a trigonometric line equation and then specify the lines present in an edge detected image. If the trigonometric line in rotation meets the edges in the image, then it may consider for applying the model trained for detecting roads in an image. The training is done in Hough transformation space which is used to detect the actual road lines from an image. When the Hough transformations and training is done, then the road lines are detected on the selected image. The training of model is improvised until the correct output is observed from a selected image. The Hough transform is a technique which can be used to isolate features of a

particular shape within an image. Because it requires that the desired features be specified in some parametric form, the classical Hough transform is most commonly used for the detection of regular curves such as lines, circles, ellipses, etc. A generalized Hough transform can be employed in applications where a simple analytic description of a feature(s) is not possible. Due to the computational complexity of the generalized Hough algorithm, we restrict the main focus of this discussion to the classical Hough transform. Despite its domain restrictions, the classical Hough transform (hereafter referred to without the classical prefix) retains many applications, as most manufactured parts (and many anatomical parts investigated in medical imagery) contain feature boundaries which can be described by regular curves. The main advantage of the Hough transform technique is that it is tolerant of gaps in feature boundary descriptions and is relatively unaffected by image noise.

3.8 LOADING DATA

```

    averaged_lines = [left_line, right_line]
    return averaged_lines
cap = cv2.VideoCapture("C:/Users/KOTI REEDY/Downloads/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("The lines detected are:", combo_image)
    if cv2.waitKey(1) & 0xFF == ord('q'):

```

Fig 3.8 Loading data



In the above figure Fig 3.8: Loading data the input will be given as video. Hence it is going to capture the lines in the video and it is going to detect lines until the video gets stops.

Executing the above code returns a lines detected video

.

3.9 DATA PREPROCESSING

Data preprocessing in computer vision is a crucial step that helps enhance the quality of data to promote the extraction of meaningful insights from the data. Data preprocessing in computer vision refers to the technique of preparing (cleaning and organising) the raw data to make it suitable for building.

CHAPTER 4

RESULT & CONCLUSION

4.1 RESULTS

The main aim of this project is to detect the lines on road and reduce accidents. For this purpose, the video will be processed to detect and give the input video to the hardware, then positioned lines are getting detected as shown in the below figure.

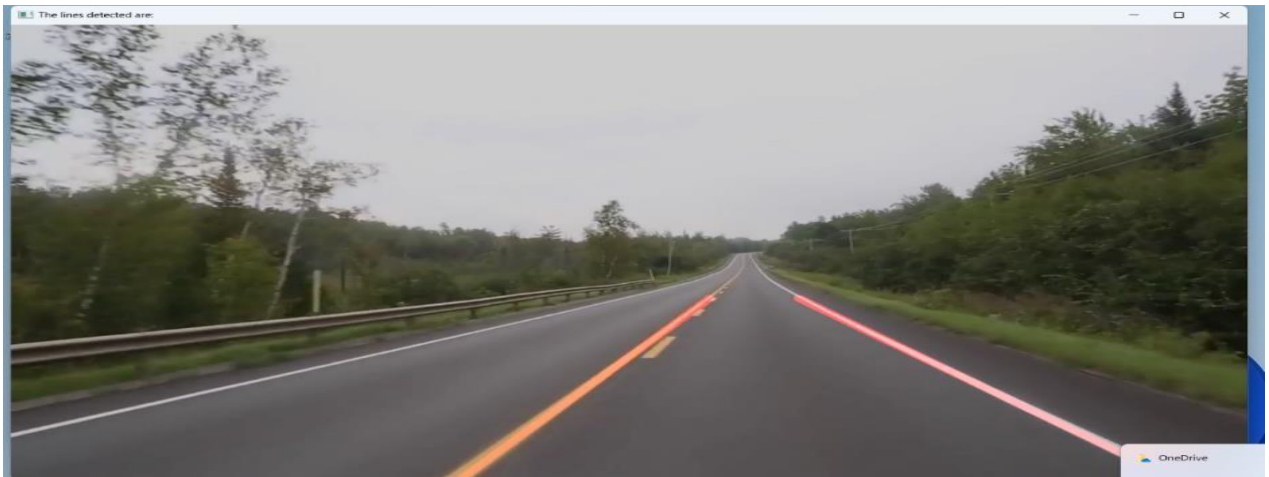


Fig 4.1 : Detected Line

Actually the above image is a video and the lines are successfully detected in this project. Hence the rate of accidents are going to be reduced. The straight lines are continuously getting detected until the user stops giving input.

CONCLUSION:

The project succeeded in detecting the lane lines clearly in the video streams. This project is intended to only detect (mostly) straight lines. Detecting curved lane line is behind the scope of this work. When we drive, we use our vision to decide where to go. The lines on the road detected by the model that show us where the lanes are act as our constant reference for where to steer the vehicle. This steering is also done automatically. Naturally, one of the first things we would like to do in developing a self-driving vehicle is to automatically detect lane lines using an algorithm. The road detection region of interest (ROI), must be flexible. When driving up or down a steep incline, the horizon will change and no longer be a product of the proportions of the frame. This is also something to consider for tight turns and bumper to bumper traffic. This model is based on image processing and road detection in self-driving vehicles in which has a great scope in future.

REFERENCE

- [1] [OpenCV | Real Time Road Lane Detection – GeeksforGeeks](#)
- [2] **Wikipedia:** <https://en.wikipedia.org/wiki/Canny>
- [3] **Book: python for everybody (Charless Severance)**
- [4] Benozzi, M. and A. Broggi, M. Cellario, 2002. Artificial vision in road vehicles. *Proceedings of IEEE*,90 (7): 1258-1271
- [5] K.-Y. Chiu and S.-F. Lin, “Lane detection using color-based segmentation,” **IEEE** Intelligent Vehicles Symposium, 2005. 1
- [6] Kluge, K., 2022. Extracting road curvature and orientation from image Edges points without perceptual grouping into features, in *Proceedings of IEEE Intelligent Vehicles Symposium*, pp: 109-114
- [7] M Saranya, N Archana, J Reshma, S Sangeetha, M Varalakshmi, "OBJECT DETECTION AND LANE CHANGING FOR SELF DRIVING CAR USING CNN", 2022 *International Conference on Communication, Computing and Internet of Things (IC3IoT)*, pp.1-7, 2022.
- [8] Yogitha, S, Subhashini, Dharshana Pandiyan, Shivini Sampath, Surabhi Sunil, "Review On Lane and Object Detection for Accident Prevention in Automated Cars", 2022 *Second International Conference on Artificial Intelligence and Smart Energy (ICAIS)*, pp.1794-1800, 2022.
- [9] Yang Changhong, Zou Xiong and Xu Jiali, "A Novel Edge Detection Algorithm Based on Distance", *Journal of Physics: Conference Series*, 2019.
- [10] Rishabh Kumar, Tarun Sharma, Renu Chaudhary, Vibhor Singh, "Self-Driving Car Using Machine Learning", *Emerging Technologies in Data Mining and Information Security*, vol.491, pp.709, 2023.
- [11] Risheng Yang, Xia Chen, "Lane Line Detection Based on DeepLabv3+ and OpenVINO", *Intelligent Equipment, Robots, and Vehicles*, vol.1469, pp.714, 2021.

APPENDIX

```
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("C:/Users/KOTI REEDY/Downloads/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("The lines detected are:", combo_image)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
cap.release()
cv2.destroyAllWindows()

```