**­­­­Vehicle Detection And Counting**

*A project report submitted in partial fulfillment of the requirements for the 7th semester of the degree of*

**BACHELOR OF TECHNOLOGY**

In

**ELECTRONICS ENGINEERING**

Submitted by

##### ARUNDHATI PORWAL (18001009011)

**BHAVNA AGRAWAL (18001009014)**

##### HARSHUL SINGLA (18001009020)

**MANTHAN PATHAK (18001009031)**

Under the Supervision of

**Mrs. NEETU GUPTA**



# Department of Electronics Engineering Faculty of Engineering and Technology

J.C. Bose University of Science and Technology, YMCA, Faridabad,

Haryana-121006

**DECEMBER, 2021**

DECLARATION

I hereby certify that the work which is being presented in this project report entitled **“VEHICLE DETECTION AND COUNTING”**, in partial fulfillment of requirements for the 7th semester of degree of BACHELOR OF TECHNOLOGY in ELECTRONICS AND INSTRUMENTAION

CONTROL ENGINEERING, submitted to the Department of Electronics Engineering, Faculty of Engineering and Technology, J.C. Bose University of Science and Technology, YMCA, Faridabad, Haryana-121006 is an authentic record of my work carried out during period from August,2021 to Decenber,2021, under the supervision of **Mrs. NEETU GUPTA**. The matter presented in this project report, has not submitted to any other University/Institute for the award of B.Tech. or any degree or diploma.

**ARUNDHATI PORWAL (18001009011)**

**BHAVNA AGRAWAL (18001009014)**

**HARSHUL SINGLA (18001009020)**

**MANTHAN PATHAK (18001009031)**

**CERTIFICATE**

This is to certify that the project entitled “*Vehicle Detection And Counting*” submitted to Department of Electronics Engineering, Faculty of Engineering and Technology, J.C. Bose University of Science and Technology, YMCA, Faridabad, Haryana-121006 by ***Arundhati Porwal, Bhavna Agrawal, Harshul Singla, Manthan Pathak*** in partial fulfillment of the requirement for the 7th semester of degree of BACHELOR OF TECHNOLOGY in ELECTRONICS INSTRUMENTAION AND CONTROL

ENGINEERING, is a bonafide work carried out by him/her under my supervision and guidance. This project work comprises of original work and has not been submitted anywhere else for any other degree to the best of my knowledge.

Project Supervisor

##### NEETU GUPTA

Assistant Professor

Department of Electronics Engineering

JCBoseUST, YMCA

Faridabad

##### Chairman

Department of Electronics Engineering

JCBoseUST, YMCA

Faridabad

# ACKNOWLEDGEMENT

We are extremely grateful to **Mrs. NEETU GUPTA** for providing us with an opportunity to work on the “Object Detection and Tracking” using Image Processing.

With the successful completion of the project under the stipulated time, I would like to take this opportunity to present my wholehearted compliments, with higher regards and warm thanks to all, who played an integral role and acted as sinew of this project.

Arundhati Porwal Bhavna Agrawal Harshul Singla Manthan Pathak

B.Tech:- 4th Year ( 7th sem EIC)

JCBUST

# Abstract

The vehicle detection system in this project is designed and implemented using Open CV library and Python. The detection and counting of vehicles is an area of focus of Intelligent transportation system (ITS), especially for traffic management. Traffic issues has become a biggest problem faced by the people for years. This rises the need for vehicle detection systems which detect the vehicles and track, classify and count then.

The object tracking algorithm used here is *centroid tracking*. The moving vehicles and their detection, tracking, and counting are very necessary for in monitoring, planning, and controlling the traffic. By analyzing the recorded video of traffic flow sequence from a camera, a video- based solution technology is developed with the help of the Open CV library for detection and counting of vehicles. Such methods helped us to detect, track and count the moving vehicles.

# Contents

1. Introduction 10

Background

Object detection and tracking Aim of Project

Motivation 2.5.Problem Statement

2.6. Approach

1. Theoretical Background 16

Image Processing Object Detection Open CV

Python NumPY Matplotlib

3.3. Object Tracking

1. Basic Concepts 24

Image Transformation Contour Detection

Color Channels Blurring Techniques

Masking

Histogram Computation Thresholding

Edge Detection

1. Vehicle Detector And Counter 37

Aim

Software Required Algorithm

Vehicle Detection Algorithm Tracking Algorithm

Final Output

1. [Result and Discussion](#_TOC_250000) 46
2. Conclusion 47
3. Application 48
4. Limitation 49
5. Future Scope 50
6. References 51

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **Figure No.** | **Title Name** | **Page No.** |
| Figure 1 | Vehicle detection in surveillance | 11 |
| Figure 2 | Flow Chart- Object Recognition | 12 |
| Figure 3 | Statistics | 14 |
| Figure 4 | Vehicle detection and counting | 15 |
| Figure 5 | Image Processing Fields | 16 |
| Figure 6 | Steps in Digital Image Processing | 18 |
| Figure 7 | Flow Chart-Steps in Digital Image Processing | 20 |
| Figure 8 | Image Recognition v/s Object Detection | 20 |
| Figure 9 | Extraction of frames from video | 23 |
| Figure 10 | Original Image | 24 |
| Figure 11 | Output Translated Image | 24 |
| Figure 12 | Rotated Image | 25 |
| Figure 13 | Resized Image | 25 |
| Figure 14 | Flipped Image | 25 |
| Figure 15 | Cropped Image | 25 |
| Figure 16 | Contours drawn | 26 |
| Figure 17 | Original Image and Gray Image | 27 |
| Figure 18 | RGB Image | 27 |
| Figure 19 | HSV Image | 28 |
| Figure 20 | Average Blur | 29 |
| Figure 21 | Guassian Blur | 29 |
| Figure 22 | Median Blur | 29 |
| Figure 23 | Bilateral Blur | 29 |
| Figure 24 | Mask | 30 |
| Figure 25 | Masked Image | 30 |
| Figure 26 | Colour Histogram | 31 |
| Figure 27 | Original Image | 32 |
| Figure 28 | Gray Image | 32 |
| Figure 29 | Simple Threshold | 33 |
| Figure 30 | Simple Threshold Inverse | 33 |
| Figure 31 | Adaptive Thresholding | 34 |
| Figure 32 | Original Image | 35 |
| Figure 33 | Gray Image | 35 |
| Figure 34 | Laplacian Edge Detector | 35 |
| Figure 35 | Sobel Edge Detector | 36 |
| Figure 36 | Canny Edge Detector | 36 |
| Figure 37 | Color to Grey conversion or masking | 37 |
| Figure 38 | Contouring | 39 |
| Figure 39 | Extraction of ROI | 39 |
| Figure 40 | Defining Boundary Boxes | 40 |
| Figure 41 | Thresholding | 40 |
| Figure 42 | Vehicle Detection | 41 |
| Figure 43 | Coordinate of tracked vehicles | 42 |
| Figure 44 | Simulation Output | 42 |
| Figure 45 | Keypoints of simulations | 43 |
| Figure 46 | Speed Estimation | 43 |
| Figure 47 | Speed Readings | 44 |
| Figure 48 | Simulation | 45 |

## Chapter 1

INTRODUCTION

**Background**

Digital Image processing is a method to process digital images to induce an enhanced image or extract some useful information from it. Nowadays digital image processing is among one of the rapidly growing technologies. We are witnessing tremendous growth with regards to AI, digital communications and visual representation thanks to the evolution of computational power, and the low cost of extremely efficient digital cameras.

In today's world, video object detection is used in a wide range of sectors. Crowd counting, vehicle number plate identification and recognition, ball tracking in sports, robotics, and traffic management are some of its applications.

The current report on ‘Object Detection and tracking using open cv and python’ based on application of image processing : ***‘Vehicle Detection And Counting.’***

Today road fatalities and traffic management are the two major issues faced by the citizens all around the country especially in urban areas. According to a survey, approximately 1.35 million people die annually as a result of road traffic crashes. Among these accidents , metropolitan cities has major contribution. So to manage traffic both manually and automatically a proper vehicle detection system is needed.

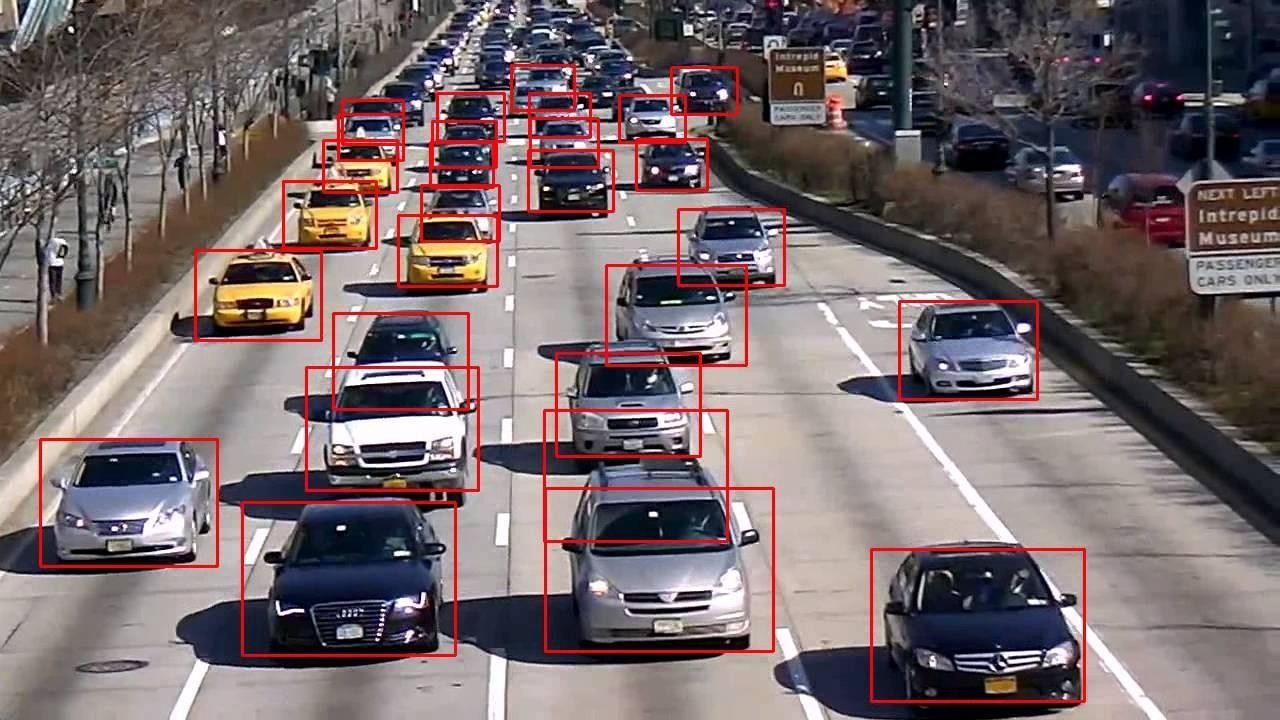


Fig 1. Vehicle Detection in surveillance system

Under the Intelligent Transportation System (ITS),the researchers try to use the intelligence to attain manifold and noble goals including safety, efficiency, environmental preservation by reducing vehicle emissions etc.. One of these genuine software is video-based vehicle detection and number plate detection, which is the project presented in this report.

Introduction to object detection and tracking

**Image Classification** predict the kind of an object in a picture or image. The input is an image with one single object such as a photograph , the output as a category of class label that is the integers mapped to the class label.

**Object Localization** detects the existence of items in a picture and uses a bounding box to represent their location. An image with one or more items, such as a photograph, is used as the input.

**Object detection** locates the existence of items in a photograph or image using a bounding box and the types or classes of the objects found. An image with one or more items, such as a photograph, is used as the input. One or more bounding boxes (e.g., defined by point,

width, and height coordinates) and a category name for each bounding box are produced.

On further extension to this breakdown of computer vision tasks is object segmentation, also called “object instance segmentation” , which means portioning of image into multiple segments ,where the recognized objects are indicated by highlighting the particular pixels of the object rather than a course of bounding box.

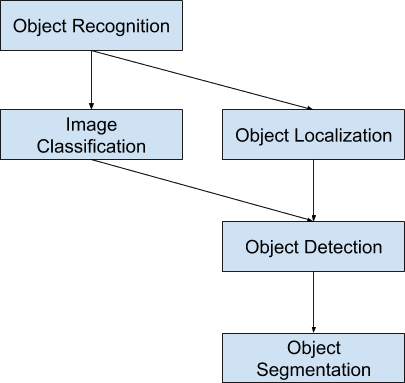


Fig 2. Flow Chart – Object Recognition

There is a difference between object detection and object tracking. Object detection is done on each single frame even in a video it detect object frame after frame. While object tracking is keeping track of the object detected during all the frames to own a history of where that object was over time.

In this project I have developed a Vehicle detector system which detects a vehicle and count the total number of vehicles using python and the open cv library. OpenCV is like a cross-platform library using which we can develop real-time computer vision applications. It primarily focuses on image processing, video capture, and analysis, which includes characteristics like as face and object detection.

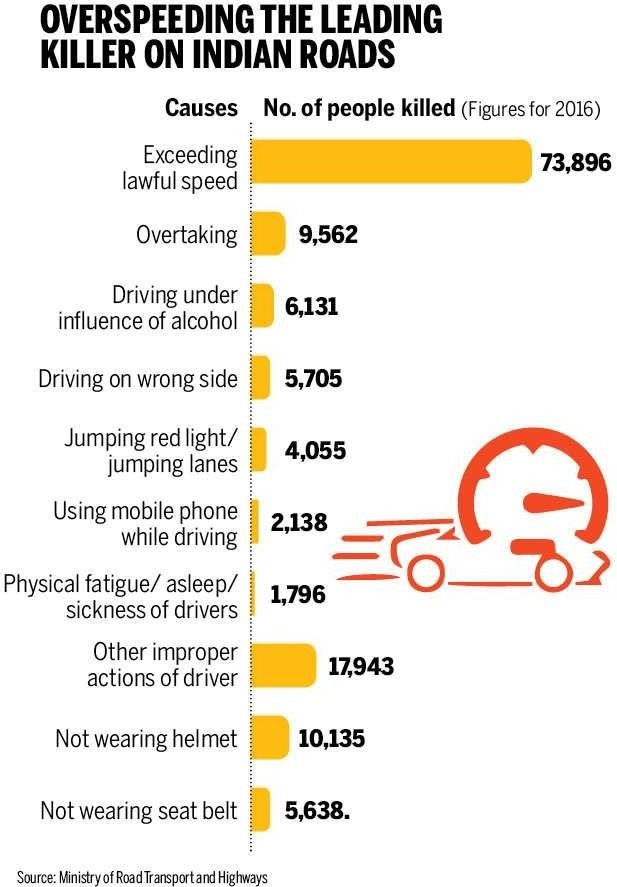
#### Aim of Project

The aim of the project is to develop a vehicle detection system which keeps the track of the vehicle, count the number of vehicles passing through a particular region and also detect the speed of vehicle. It uses a stable camera for object detection that is the background does not change with time and only the vehicles move. This project uses python and Open cv library which is a cross-platform library using which we can develop real- time computer vision applications.

#### Motivation

Traffic counts and vehicle tracking are critical bits of data for a variety of transportation applications, from transportation planning to modern intelligent transportation systems. The road fatalities and traffic management are the two major issues faced by many citizens all around the country especially in urban areas. With the growing demand and recognition of autonomous vehicles, sets high requirements for pedestrian safety.

The startling losses both in human lives and finance caused by vehicle accidents. Detecting vehicles in images acquired from a moving platform could be a challenging problem.



Vehicle detection is the area of focus of Intelligence transportation system [ITS] to keep the record for enhancing public safety , reducing congestion, improved access to travel and transit information , generate cost savings to motor carriers, transit operators, toll authorities, and government agencies.

Fig 3. Statistics

This gap in precarious technique motivates to develop the proposed algorithm. A classification and counting system, such as the one presented here, can provide critical information to a decision-making agency.

#### Problem Statement

Studies have shown that almost every day 3,700 are killed globally in road traffic crashes, and over half of them are pedestrians, cyclists and motorcyclists. The World Health Organization analyse the various risk factors causing traffic accidents, and nearly (94-96%) of all car traffic accidents are due to human errors. For civilian and military applications like as highway traffic monitoring, management, and urban traffic planning, vehicle identification and tracking applications play a critical role.

#### Methodology

I will be devising a model to detect and count the vehicles which can be installed in cameras along side traffic signals , in highways etc.



Fig 4. Vehicle detection and Counting

Chapter 2

## THEORETICAL BACKGROUND

**Image Processing**

Digital image processing refers to processing of images which are digital in nature to improve the pictorial information for human perception , for image processing for autonomous machine application and for efficient storage and transmission.

##### Image Processing Fields

Computer graphics is the process of creating visuals on a computer.

* Image Processing: Image enhancement or other image manipulation
* Computer Vision: Image content analysis

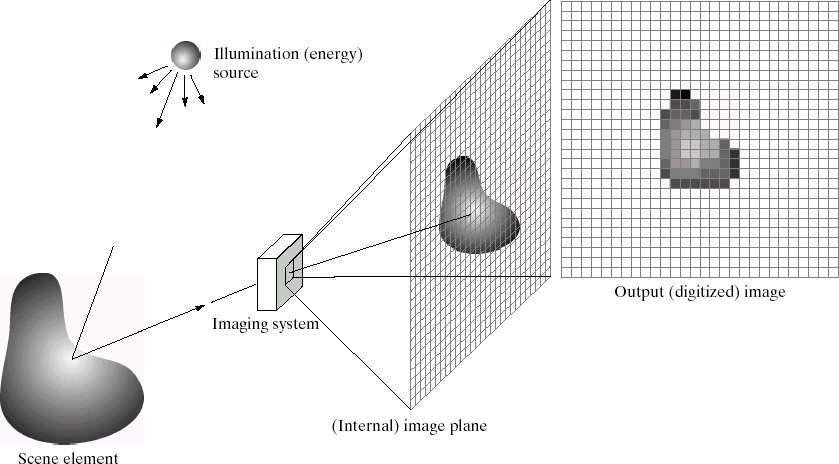


Fig 5. Image Processing Fields

##### Purpose of Image processing

The purpose of image processing are :

Visualization - Pay attention to the things that aren't visible.

Sharpening and Restoration -To make a better image, picture sharpening and restoration are used.

Image retrieval - Look for the image you're looking for.

Pattern measurement - calculates the size of distinct items in an image. Image recognition -It distinguish the objects in an image .

##### Fundamental steps in Digital Image Processing :

1. **Image Acquisition**

The first of the fundamental processes in digital image processing is this phase or procedure. It might be as simple as being provided an image that has already been converted to digital format. Pre-processing, such as scaling, is usually done during the image acquisition stage.

##### Image Enhancement

Image enhancement is one of the most straightforward and visually appealing aspects of digital image processing. Enhancement techniques are used to bring out detail that has been concealed or to simply highlight certain features of interest in an image. Changing the brightness and contrast, for example.

##### Image Restoration

Image restoration is a field that focuses on enhancing the appearance of an image. Picture restoration, unlike enhancing, is objective in the sense that restoration approaches are usually based on mathematical or probabilistic models of image degradation.

##### Color Image Processing

Due to the extremely considerable expansion in the use of digital photographs over the Internet, color image processing has become increasingly important. Color modelling and processing in the digital environment are examples of this.

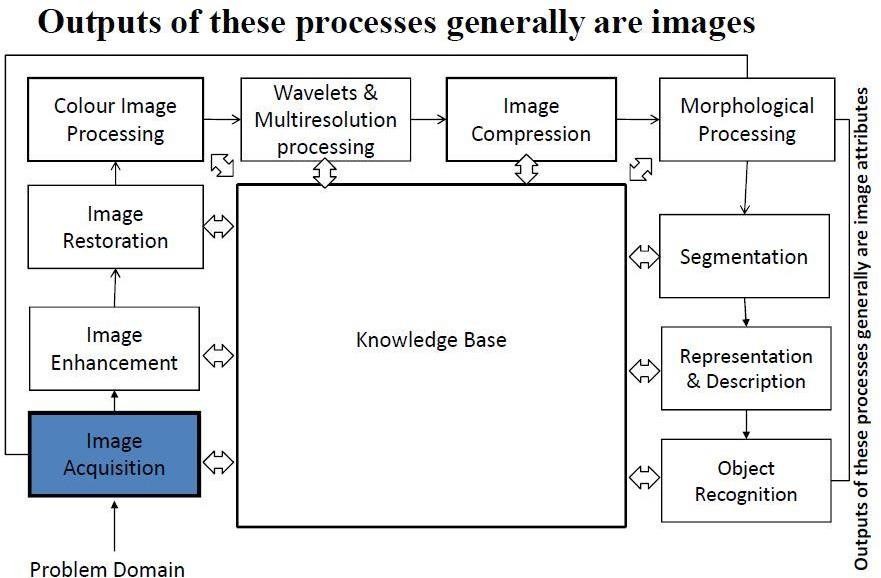


Fig 6. Steps in Digital Image Processing

##### Wavelets and Multiresolution Processing

Wavelets are the building blocks for representing images at varying resolutions. For data compression and pyramidal representation, images are successively subdivided into smaller areas.

##### Compression

Compression refers to methods for minimizing the amount of storage or bandwidth necessary to save or transmit an image. Compression of data is extremely important, especially while using the internet.

##### Morphological Processing

Morphological processing is concerned with tools for extracting picture components that can be used to represent and describe shape.

##### Segmentation

Segmentation is a technique for dividing a picture into its component components or objects. In general, one of the most difficult tasks in digital image processing is autonomous segmentation. A robust segmentation approach takes the process a long way toward solving image challenges that need individual object identification.

##### Representation and Description

The output of a segmentation stage, which is normally raw pixel data, is nearly always followed by representation and description, forming either the region's boundary or all of the points within the region. Choosing a representation is simply one aspect of the process of translating raw data

into a format that can be processed by a computer. Extracting qualities that result in quantitative information of interest or are fundamental for distinguishing one class of objects from another is the subject of description

##### Object recognition

Recognition is the process of assigning a label to an object based on its descriptors, such as "car."

##### Knowledge Base:

Knowledge can be as easy as identifying portions of an image where the information of interest is known to be located, hence reducing the amount of time spent searching for it. An interconnected list of all main conceivable flaws in a materials inspection problem, or an image database containing high-resolution satellite photos of a region in association with change- detection applications, are examples of knowledge bases.

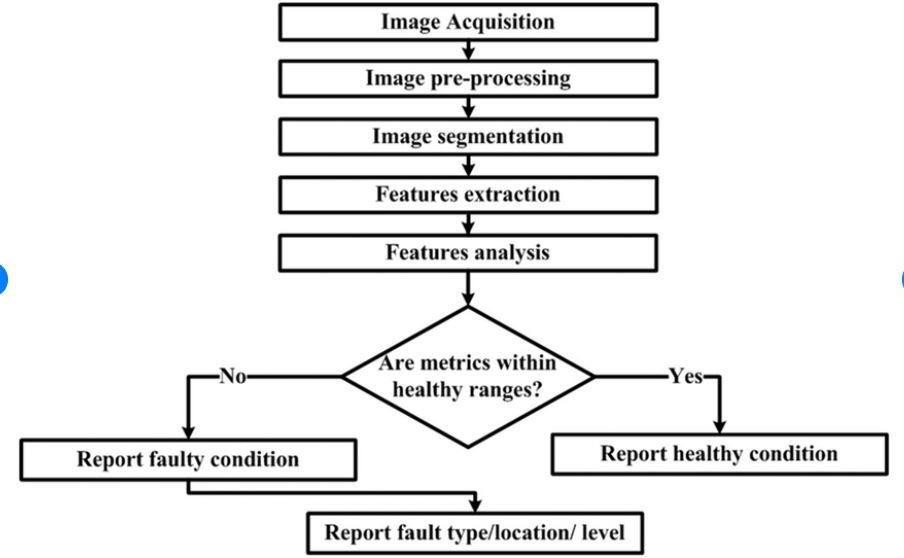


Fig 7. Flow Chart –Steps in Digital Image Processing

### Object Detection

Object detection is a computer vision approach that allows a system to recognise and find things in images or videos. The segmentation's bounded box, as illustrated in the figure, can assist a person or driver in identifying and locating items in a matter of seconds, much faster than unprocessed photos. As a result, object detection's purpose is to transfer this intelligence to a computer.

Object detection can be accomplished using a variety of methods. More information on the open cv library and python is provided below to further clarify object detection. The difference between picture recognition and object detection is depicted in the diagram below.

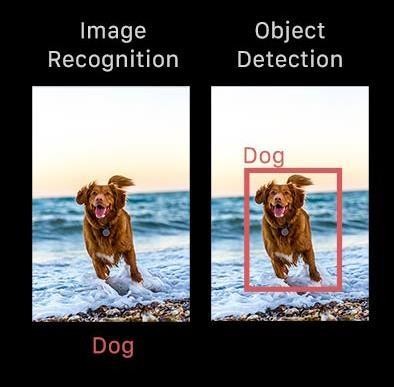


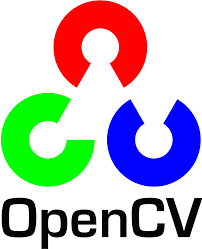
Fig 8. Image recognition v/s Object Detection

Object detection algorithms can be used to do extremely important high-value activities such as surveillance, traffic management, counting the number of objects, determining the relative size of the objects, evaluating distance between the objects, and so on.

In this project I have used Open cv library in python to perform the detection of vehicles and count them . More knowledge on the open cv library is presented below.

### Open CV Library

OpenCV (Open Source Computer Vision Library)



is a large open source library for computer vision, machine learning, and image processing, and it currently plays a key part in real-time operations, which are critical in today's systems.

OpenCV was created to provide a standardised infrastructure for computer vision applications and to help commercial goods incorporate machine perception more quickly. Because OpenCV is a BSD-licensed product, it is simple for businesses to use and alter the code. More than 2500 optimised algorithms are included in the library, which contains a comprehensive mix of both classic and cutting-edge computer vision and machine learning techniques. These techniques can be used to distinguish and detect faces, identify objects, classify human behaviours in recordings, monitor camera movements, track moving objects, extract 3D models of objects, and generate 3D point clouds from stereo images cameras, stitch images together to make a high quality image of an entire area, find comparable images from an image database, eliminate red eyes from flash images, track eye movements, detect scenery and create markers to overlay it with augmented reality, and so on.

### Python

Python is a dynamically semantic, interpreted, and object-oriented high-level programming language. Because of its high-level built-in data structures, dynamic typing, and dynamic binding, it's perfect for Rapid Application Development and as a scripting or glue

language for integrating existing components. Python's concise, easy-to-learn syntax prioritises readability, which lowers software maintenance costs. Modules and packages are supported by Python, which fosters programme modularity and code reuse. The Python interpreter and its substantial standard library are available to install and distribute in source or binary form for all major platforms.

### Numpy

NumPy is a Python library that enables users to interact with arrays. It also provides functions for working with matrices, fourier transforms, and linear algebra. We have lists in Python that act as arrays, however they are slow to process. NumPy intends to deliver a 50-fold quicker array object than ordinary Python lists.

The array object in NumPy is named ndarray, and it comes with a slew of helper functions to make working with it a breeze. In data research, when speed and resources are critical, arrays are widely employed. NumPy arrays, unlike lists, are stored in a single continuous location in memory, allowing programmes to access and manipulate them quickly. In computer science, this is referred to as locality of reference. This is the primary reason why NumPy excels lists. It's also been modified to work with the most recent CPU architectures.

### Matplotlib

Matplotlib is a data visualisation and graphical plotting package for Python and its numerical extension NumPy that runs on all platforms. As a result, it provides an open source alternative to MATLAB. Matplotlib's APIs (Application Programming Interfaces) can also be used to incorporate charts in graphical user interfaces.

In most cases, a Python matplotlib script is constructed so that only a few lines of code are required to generate a visual data plot. Two APIs are overlaid by the matplotlib scripting layer:

* The pyplot API is a network of Python code objects, with matplotlib.pyplot at the top
* An OO (Object-Oriented) API collection of items that can be put together more easily than pyplot. This API allows you to use Matplotlib's backend layers directly.

### Object Tracking

Object detection and tracking are two different things. In object detection, we identify the object in every frame and frame after frame, however in object tracking, we maintain track of where the object is at all times. The project I made detects objects (cars) using a steady camera and keeps track of how many vehicles pass through a given area.

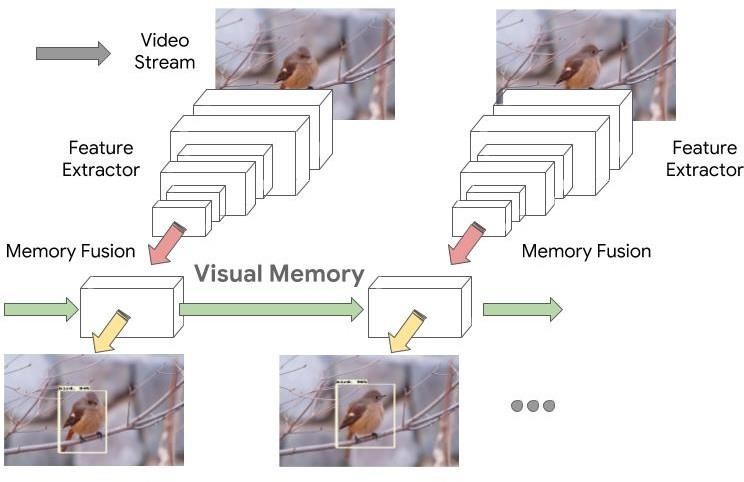


Fig 9. Extraction of frames from video

Chapter 3

## Basic Concepts

Some basics concepts regarding image processing in python using open cv library are described below.

**Image Transformation**

[Image transformation](https://en.wikipedia.org/wiki/Transformation_matrix) is a coordinate changingfunction, it maps a (x, y) points in one coordinate system to points (x', y') in another coordinate system. An image is selected and image transformations are applied on this given image .

1. **Translation –** Shifting image along x, y axis.

*# -x --> Left # -y --> Up*

*# x --> Right # y --> Down*

def translate(img, x, y):

transMat = np.float32([[1,0,x],[0,1,y]]) dimensions = (img.shape[1], img.shape[0])

return cv.warpAffine(img, transMat, dimensions translated = translate(img, -100, 100) cv.imshow(**'Translated'**, translated)



Fig 10. Original Image

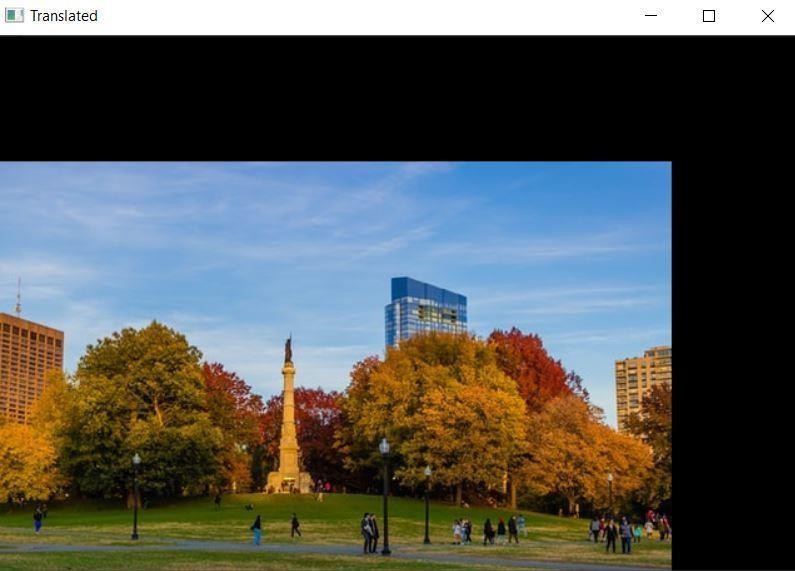


Fig 11 . Output Translated Image

1. **Rotation**

**def** rotate(img, angle, rotPoint=**None**):

(height,width) = img.shape[:2]

**if** rotPoint **is None**:

rotPoint = (width//2,height//2)

rotMat = cv.getRotationMatrix2D(rotPoint, angle, 1.0) dimensions = (width,height)

**return** cv.warpAffine(img, rotMat, dimensions) rotated = rotate(img, -45)

cv.imshow(**'Rotated'**, rotated) Fig 12. Rotated image

1. **Resizing**

resized = cv.resize(img, (500,500), interpolation=cv.INTER\_CUBIC) cv.imshow(**'Resized'**, resized)

1. **Flipping**

flip = cv.flip(img, -1) cv.imshow(**'Flip'**, flip)

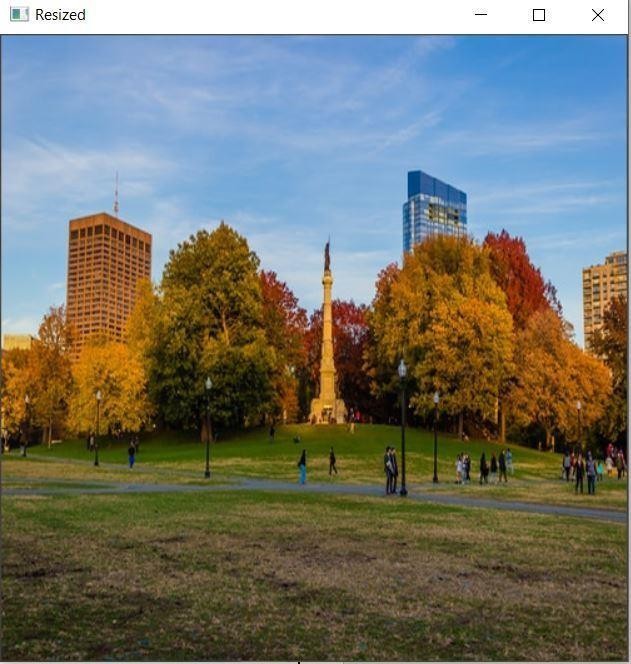


Fig 13. Resized Image Fig. 14 Flipped Image

##### Croppping

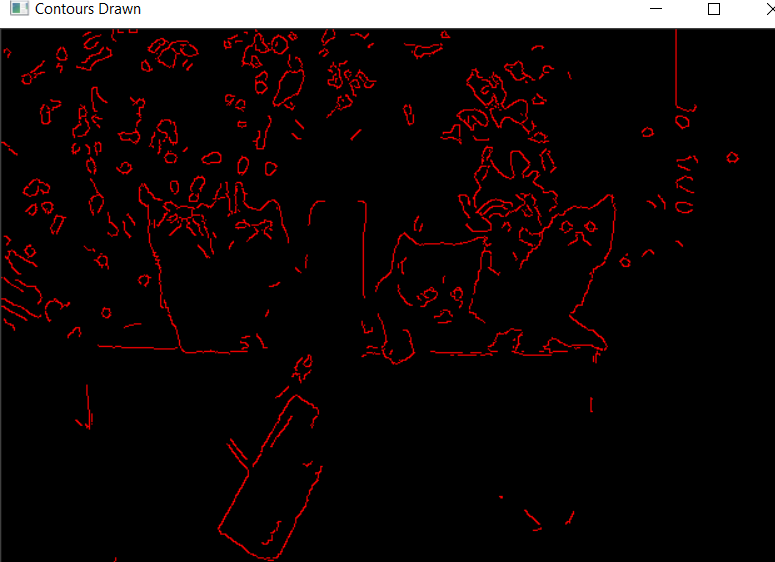
cropped = img[200:400, 300:400] cv.imshow(**'Cropped'**, cropped)

Fig 15. Cropped Image

**Contour Detection**

Contour detection is a procedure that can be described as a curve that connects all continuous points (along with the boundary) which have the same colour or intensity. The method not only finds image edges, but also arranges them in a hierarchy. The following image was used to detect contours.



img = cv.imread(**'cats.jpg'**) cv.imshow(**'Cats'**, img)

blank = np.zeros(img.shape, dtype=**'uint8'**) cv.imshow(**'Blank'**, blank)

gray = cv.cvtColor(img, cv.COLOR\_BGR2GRAY) cv.imshow(**'Gray'**, gray)

blur = cv.GaussianBlur(gray, (5,5), cv.BORDER\_DEFAULT)

cv.imshow(**'Blur'**, blur)

canny = cv.Canny(blur, 125, 175) cv.imshow(**'Canny Edges'**, canny)

contours, hierarchies = cv.findContours

(canny, cv.RETR\_LIST, cv.CHAIN\_APPROX\_SIMPLE)

print(**f'{**len(contours)**} contour(s) found!'**) Fig 16. Contours Drawn

cv.drawContours(blank, contours, -1, (0,0,255), 1) cv.imshow(**'Contours Drawn'**, blank)

cv.waitKey(0)

### Color Channels

Color spaces are a technique of representing the colour channels in an image that give it its distinct hue. There are a number of different colour spaces, each with its own meaning. RGB (Red, Green, Blue), CMYK (Cyan, Magenta, Yellow, Black), HSV (Hue, Saturation, Value), and others are some of the most used colour spaces.

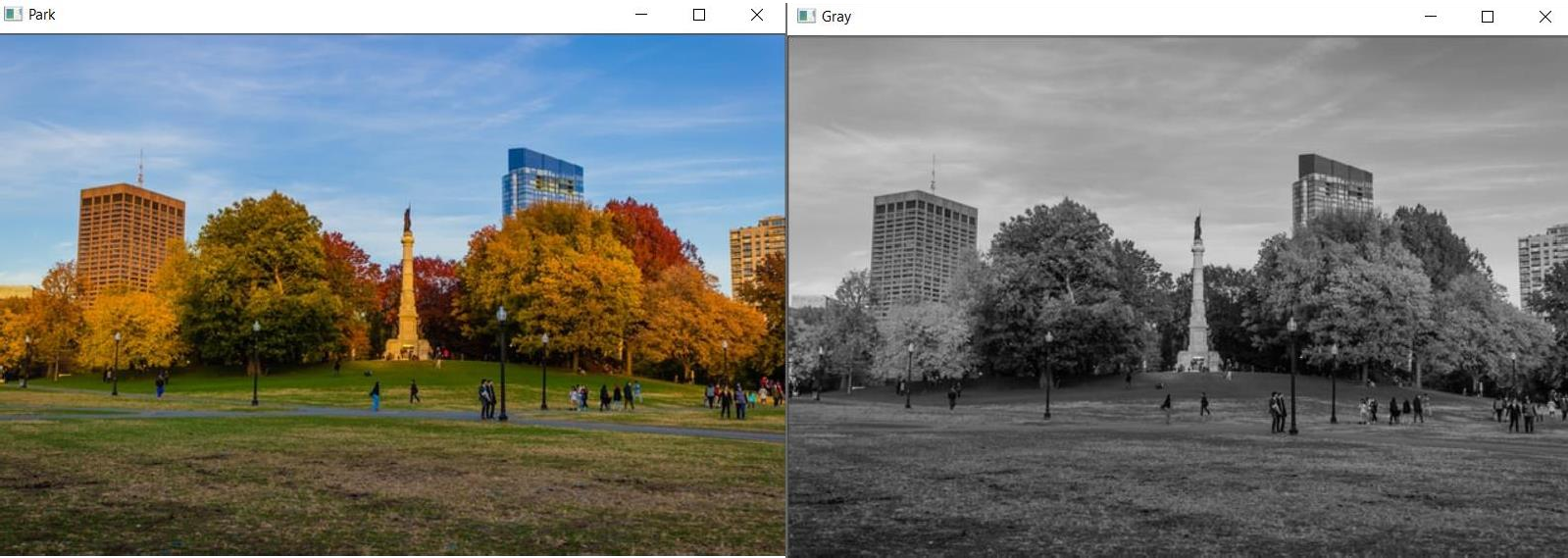


Fig 16. Original Image and Gray Image

gray = cv.cvtColor(img, cv.COLOR\_BGR2GRAY) cv.imshow(**'Gray'**, gray)

**BGR color space:** RGB is OpenCV's default colour space. It does, however, store colour in the BGR format. It's an additive colour model in which the various intensities of Blue, Green, and Red produce various hues of colour.

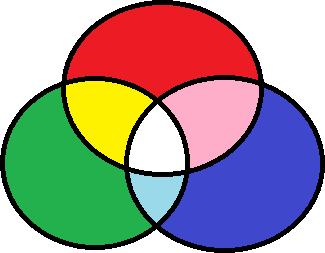
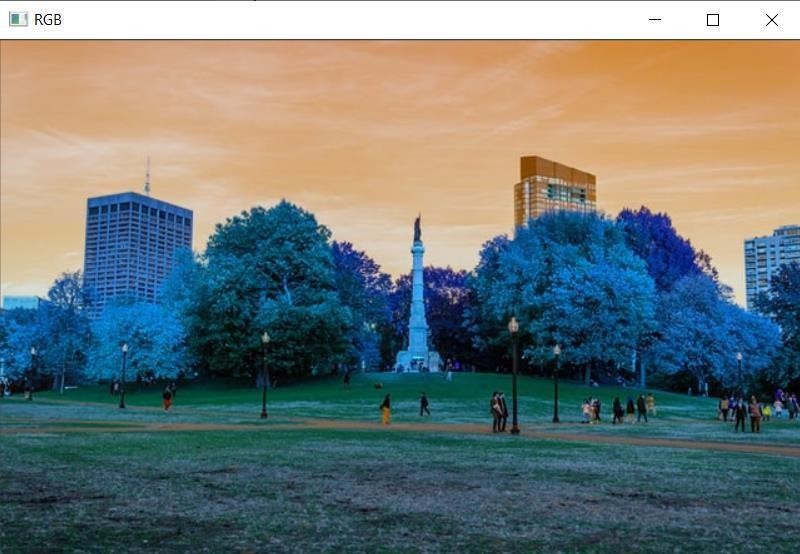
 

Fig 18 . RGB image

rgb = cv.cvtColor(img, cv.COLOR\_BGR2RGB) cv.imshow(**'RGB'**, rgb)

**HSV color space:** It uses a cylindrical representation of RGB colour points to store colour information. It tries to capture the colours as seen by the human eye. The hue ranges from 0-179, the saturation ranges from 0-255, and the value ranges from 255. It's primarily used for colour segmentation.



Fig 19. HSV Image

hsv = cv.cvtColor(img, cv.COLOR\_BGR2HSV) cv.imshow(**'HSV'**, hsv)

### Blurring Techniques

One of the most significant pre-processing phases in computer vision and image processing is smoothing and blurring. We can lower the amount of high-frequency material, such as noise and edges (i.e., an image's "detail") by smoothing a picture before applying algorithms like edge detection or thresholding.

Surprisingly, lowering the clarity in an image or blurring the image makes it easier to detect specific items. This also permits us to emphasis on the image's larger structural elements .

The four main smoothing and blurring options are:

**Simple average blurring Median filtering**

**Gaussian blurring Bilateral filtering**



Fig 20.Average Blur Fig 21. Gaussian Blur

*# Averaging*

average = cv.blur(img, (3,3)) cv.imshow(**'Average Blur'**, average)

*# Gaussian Blur*

gauss = cv.GaussianBlur(img, (3,3), 0) cv.imshow(**'Gaussian Blur'**, gauss)

*# Median Blur*

median = cv.medianBlur(img, 3) cv.imshow(**'Median Blur'**, median)

*# Bilateral*

bilateral = cv.bilateralFilter(img, 10, 35, 25)cv.imshow(**'Bilateral'**, bilateral)

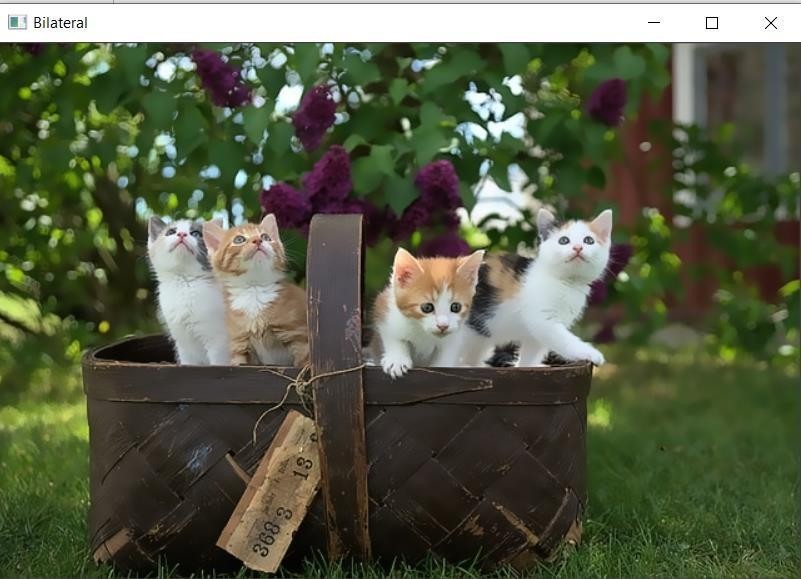


Fig 22. Median Blur Fig 23. Bilateral Blur

### Masking

Using Bitwise operators, we can extract regions from photos that are fully arbitrary in shape. A mask helps us to concentrate just on the areas of the image that we are interested in.

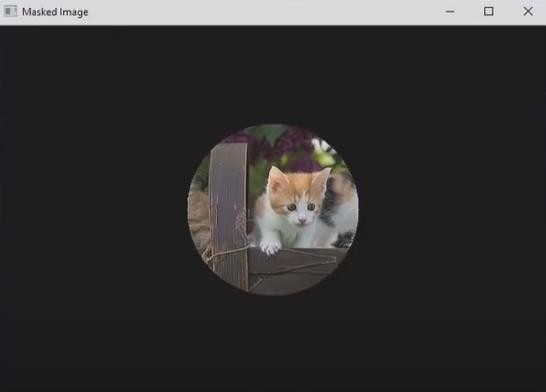
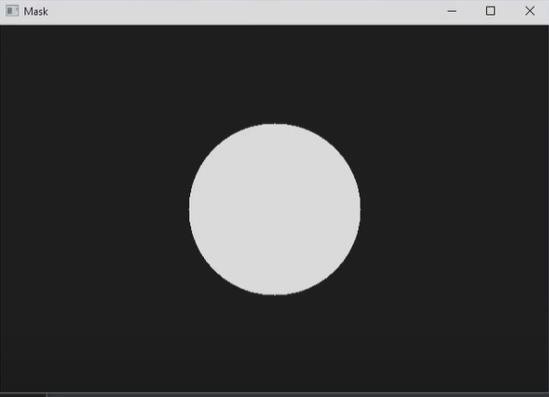


Fig 24. Mask Fig 25. Masked Image

img = cv.imread(**'cats.jpg'**) cv.imshow(**'Cats'**, img)

blank = np.zeros(img.shape[:2], dtype=**'uint8'**) mask = cv.circle(blank, (img.shape[1]//2, img.shape[0]//2) , 100, 255, -1)

masked = cv.bitwise\_and(img,img,mask=mask) cv.imshow(**'Mask'**, masked)

### Histogram Computation

In image processing, a histogram is a crucial tool. It's a graphical representation of how data is distributed. An image histogram is a graphical depiction of a digital image's pixel intensity distribution.

The x-axis represents the variable's possible range of values. This range is broken into bins, which are intervals of time. The number of values that fall within that interval is represented on the y-axis. The pixel intensity is

plotted on the X-axis, while the frequency is plotted on the Y-axis. We can choose how many bins to utilise.

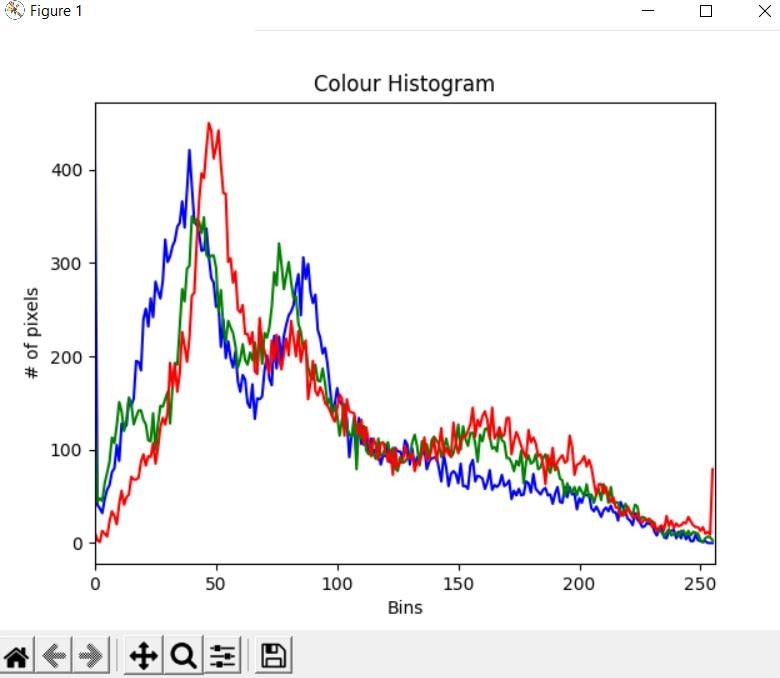
Using Bitwise operators, we can extract regions from photos that are fully arbitrary in shape. For the gray scale we have a single channel, hence a single histogram. For colored image we have 3 channels, hence 3 histograms. Histogram calculation of an image is very useful as it gives an intuition regarding some properties of the image like the **tonal range**, the **contrast** and the **brightness**.

##### Code And Output:

img = cv.imread(**'cats.jpg'**) cv.imshow(**'Cats'**, img)

blank = np.zeros(img.shape[:2], dtype=**'uint8'**) mask = cv.circle(blank, (img.shape[1]//2, img.shape[0]//2) , 100, 255, -1)

masked = cv.bitwise\_and(img,img,mask=mask) cv.imshow(**'Mask'**, masked)

*# Colour Histogram*

plt.figure()

plt.title(**'Colour Histogram'**) plt.xlabel(**'Bins'**) plt.ylabel(**'# of pixels'**) colors = (**'b'**, **'g'**, **'r'**)

**for** i,col **in** enumerate(colors): hist = cv.calcHist([img], [i], mask, [256], [0,256])

plt.plot(hist, color=col) plt.xlim([0,256])

plt.show() cv.waitKey(0)

Fig 26. Colour Histogram

### Thresholding

Thresholding is one of the most common segmentation techniques in computer vision and it allows us to separate the *foreground* (i.e., the objects that we are interested in) from the *background* of the image.

Thresholding comes in two forms:

* 1. **Static thresholding** , in which we manually give parameters to segment the image, works exceptionally well in regulated lighting settings where we can ensure a high contrast between the foreground and background of the image.
  2. **Adaptive thresholding** , which rather than trying to threshold an image globally with a single value, splits it down into smaller pieces and thresholds each one separately and individually.

**Code and Output**

**Import** cv2 **as** cv

img = cv.imread(**'../Resources/Photos/cats.jpg'**) cv.imshow(**'Cats'**, img)

gray = cv.cvtColor(img, cv.COLOR\_BGR2GRAY) cv.imshow(**'Gray'**, gray)



Fig 27. Original Image Fig 28. Gray Image

*# Simple Thresholding*

threshold, thresh = cv.threshold(gray, 150, 255, cv.THRESH\_BINARY ) cv.imshow(**'Simple Thresholded'**, thresh)



Fig 29. Simple Threshold

threshold, thresh\_inv = cv.threshold(gray, 150, 255, cv.THRESH\_BINARY\_INV )

cv.imshow(**'Simple Thresholded Inverse'**, thresh\_inv)

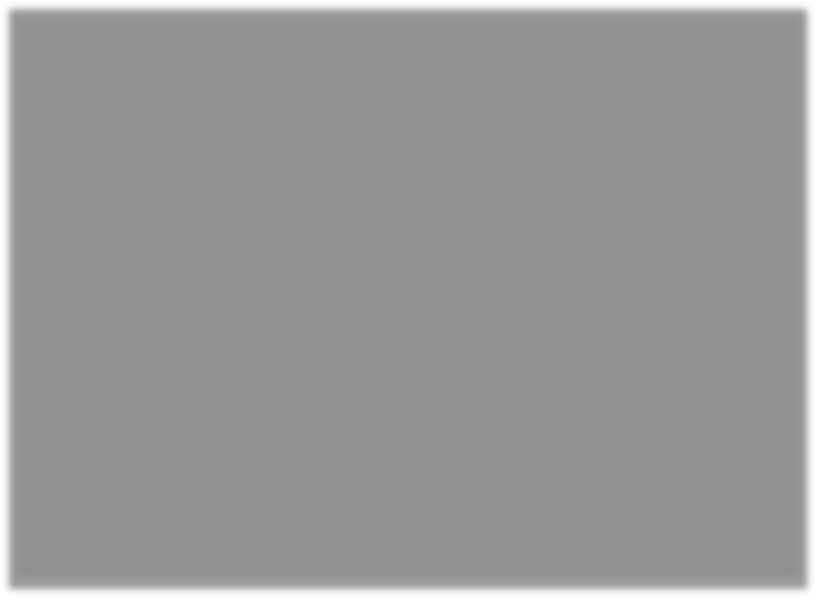


Fig 30. Simple Threshold Inverse

*# Adaptive Thresholding*

adaptive\_thresh = cv.adaptiveThreshold(gray, 255, cv.ADAPTIVE\_THRESH\_GAUSSIAN\_C, cv.THRESH\_BINARY\_INV, 12, 10)

cv.imshow(**'Adaptive '**, adaptive\_thresh) cv.waitKey(0)

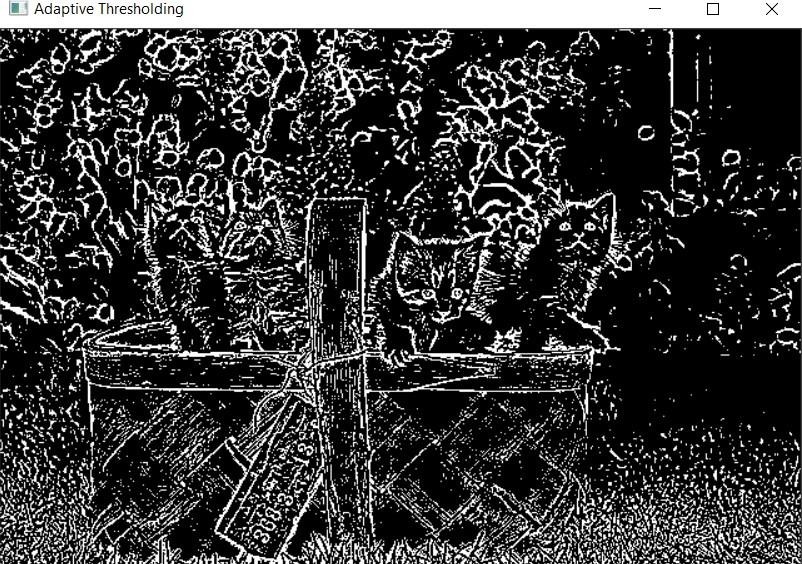


Fig 31. Adaptive Thresholding

### Edge detection

**Laplacian edge detector** - The second derivatives of a picture are compared via a Laplacian edge detector. It is a metric for how quickly the first derivative changes in a single pass. Laplacian edge detection uses one kernel and contains negative values in a very cross pattern.

**Sobel edge detector** could be a gradient based method that is based on the first order derivatives. It computes the image's principal derivatives individually for the X and Y axes. Unlike the Sobel edge detector, the Laplacian edge detector uses just one kernel.

**Canny edge detector** The goal of the Canny edge detector is to meet three basic criteria:

Low error rate: This refers to the ability to detect only existing edges.

Good localization: The distance between detected edge pixels and real edge pixels must be kept to a minimum.

Minimal response: Only one detector response per edge is required.

**CODE AND OUTPUT**

img = cv.imread(**'parki.jpg'**) cv.imshow(**'Park'**, img)

gray = cv.cvtColor(img, cv.COLOR\_BGR2GRAY) cv.imshow(**'Gray'**, gray)

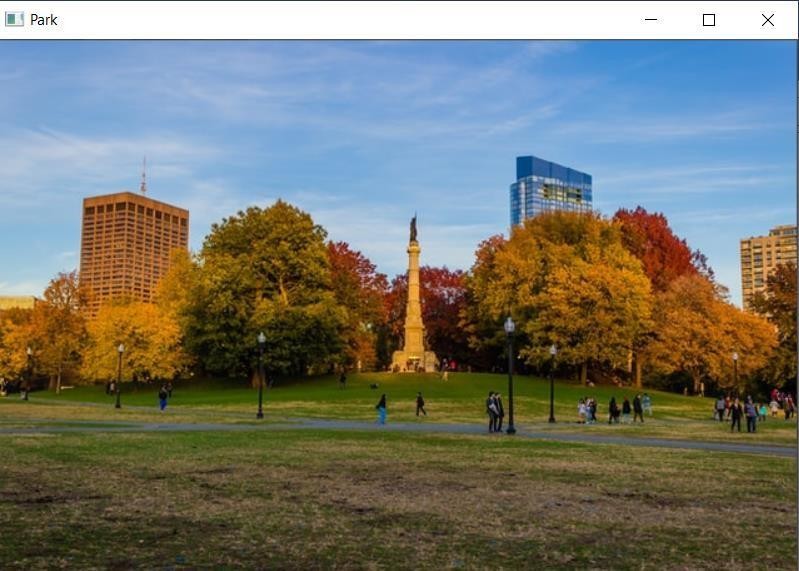


Fig 32. Original Image Fig 33, Gray Image

*# Laplacian*

lap = cv.Laplacian(gray, cv.CV\_64F) lap = np.uint8(np.absolute(lap)) cv.imshow(**'Laplacian'**, lap)

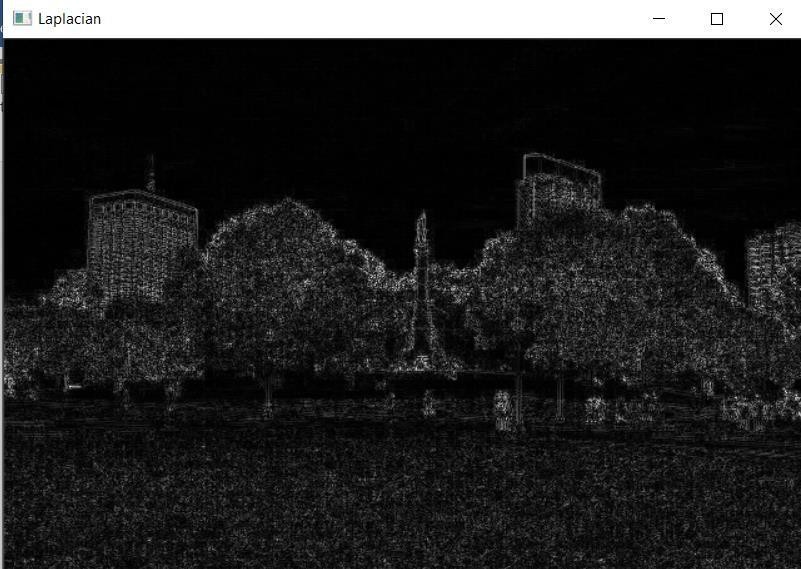


Fig 34. Laplacian Edge Detector

*# Sobel*

sobelx = cv.Sobel(gray, cv.CV\_64F, 1, 0)

sobely = cv.Sobel(gray, cv.CV\_64F, 0, 1) combined\_sobel = cv.bitwise\_or(sobelx, sobely) cv.imshow(**'Combined Sobel'**, combined\_sobel)



Fig 35. Sobel Edge Detector

*# Canny*

canny = cv.Canny(gray, 150, 175) cv.imshow(**'Canny'**, canny) cv.waitKey(0)

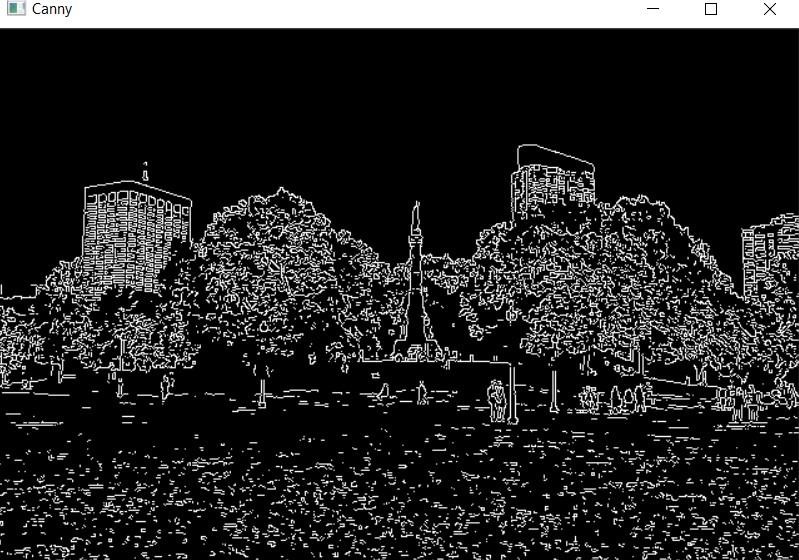


Fig 36. Canny Edge Detector

Chapter 4

## Vehicle Detector and Counter

Aim : To design a vehicle detector and counter using open CV library in python. Along with it detecting the speed of vehicles in maintaining the database.

Software Required : Python 3.9, IDE – Py charm, Open cv library installed.

**Object Detection and Tracking**

Object detection is detecting any object in every single frame and frame after frame while tracking also does the same but it also keeps the history of where the object is at a time after time.

Object tracking involves the following steps:

1. First of all, it takes an initial set of object detections that is the input set of bounding coordinates.
2. It gives a unique Id for each detection which makes it possible for us to count the objects in a video.
3. Now each of the objects is tracked as it moves around frames in a video, maintaining their unique Id.
4. Then we have calculated the speed of all the vehicles and maintained a database of this data.

In this Project I have used **Centroid Tracking Method** to track the vehicles and keep the count of vehicles as it passes from the region of interest.

Algorithm

This project is divided into two parts :

1. Vehicle detection
2. Vehicle tracking

##### Vehicle detection

There are the following steps involved in detecting an object:

1. Import a video file(.mp4) and show the frame .
2. Using the function background subtractor we get the background ratio and create a mask that is the objects needed will be white and the rest black.



Fig 37. Color to grey conversion or masking

1. Remove the noise from the video by using smoothing techniques.
2. Now find and draw the contours of the boundary. For improving the extraction all the smaller elements were removed by focussing attention on objects that are larger than a certain area.



Fig 38. Contouring

1. In this project, I have defined a region of interest that is from which position we want to extract the frame and detected the vehicles passing through that area.

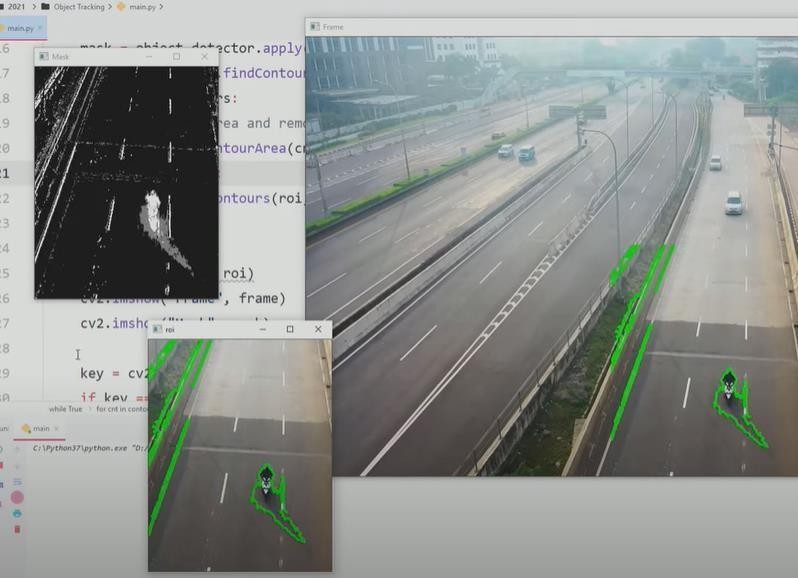


Fig 39. Extraction of ROI

1. Apply mask on the ROI region.

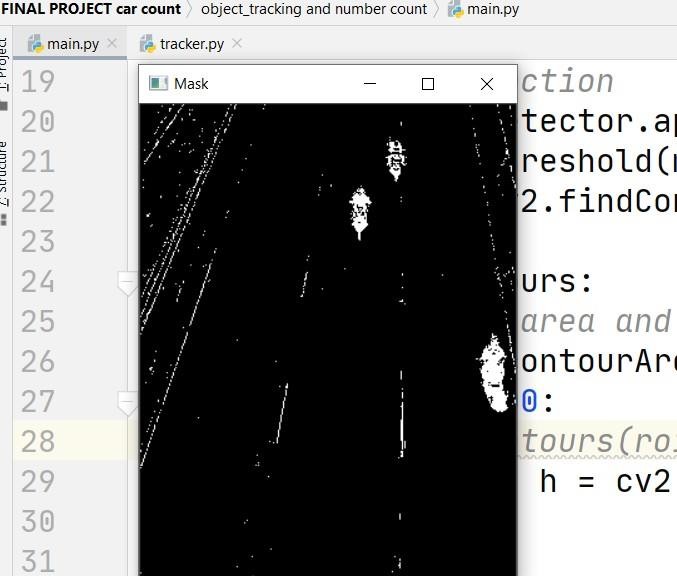


Fig 40 Defining boundary boxes

1. Draw rectangles on ROI.
2. This was detecting objects as well as their shadows, so to remove it we give a certain threshold value to the mask this is called thresholding.

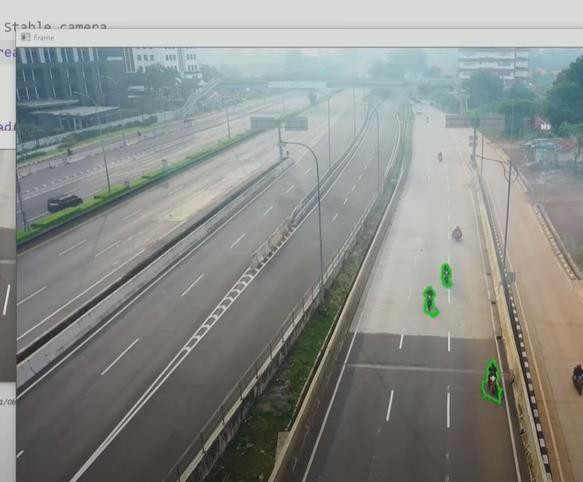


Fig 41 Thresholding

1. Now all the whites from the shadow are removed and perfect vehicles are detected.

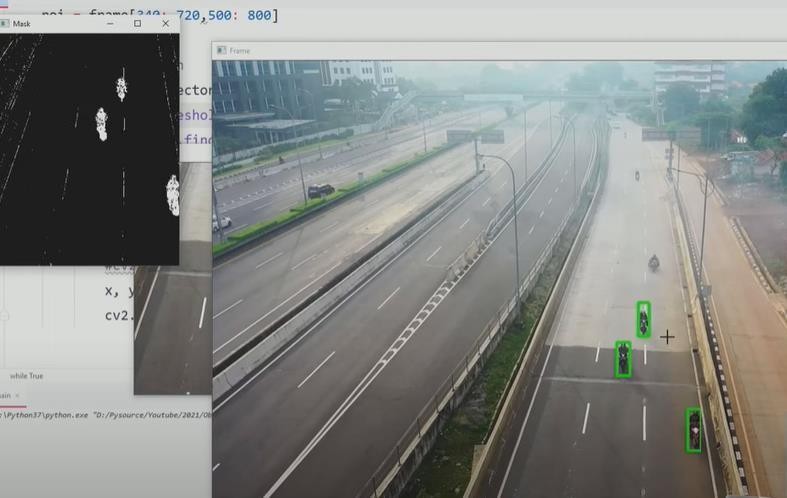


Fig 42 Vehicle detection

After these steps first half of the project that is Vehicle Detection is completed in which on a video file we performed certain operations like noise reduction, smoothing, edge detection, background subtraction, finding contours, defining a region of interest (ROI), applying threshold value, and drawing rectangles on the detected vehicle.

##### Vehicle Tracking

In the first half, we have detected the vehicle but have no clue where the object was before so for this, we do vehicle tracking.

1. Create a tracker file using the Centroid Tracking Algorithm and import it in the main file.
2. Create a tracker object this will take all the bounding boxes of the object and insert them in an array. The array contains the coordinates (x,y) and width and height (w,h) of the bounding box

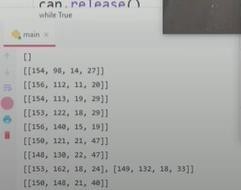


Fig 43. Coordinate of tracked vehicles

1. After presenting the result the lanes that pass through out ROI are identified and their positions inserted in a specific array. Obviously, the more the vehicles identified the larger our array will be.

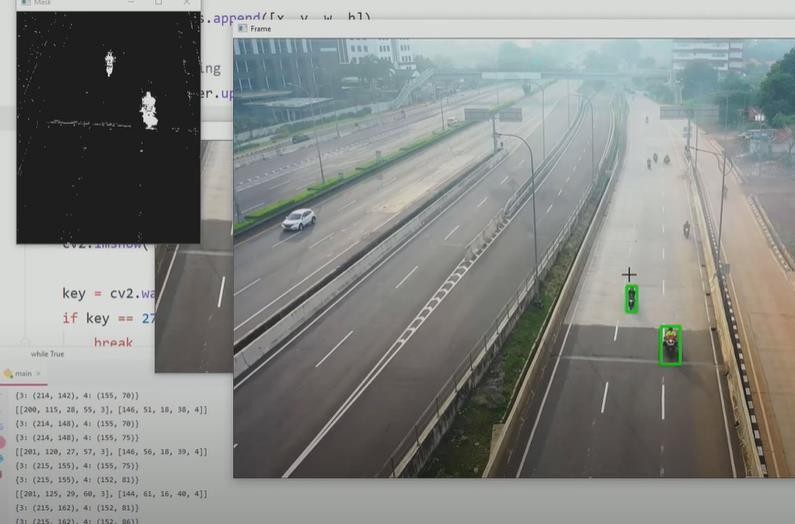


Fig 44. Simulation output

1. Associating unique IDs: This detected array is passed to the tracker. update() and in return, we get an array and since the tracker gives the history, so a unique Id will also be assigned for each object.



Fig 45. Keypoints of simulations

1. For reference, we are taking two regions in the ROI, when the vehicle reaches to the first region, our timer starts and when the vehicles reaches second region, it stops. This way, we are getting time taken by our vehicle to cross the specified portion of ROI.
2. We have observed the distance between the two reference regions. Now using the fundamental formula of speed(speed=distance/time) we can calculate the speed.



Fig 46. Speed Estimation

1. The last step is to put text on the bounding objects (vehicles).
2. We are saving cars image in the jpeg format to differentiate them according to their speed.
3. Also we are maintaining a text document for the speed of all the vehicles. Also it shows the total number of vehicle detected as well as which vehicle is exceeding the speed limit.

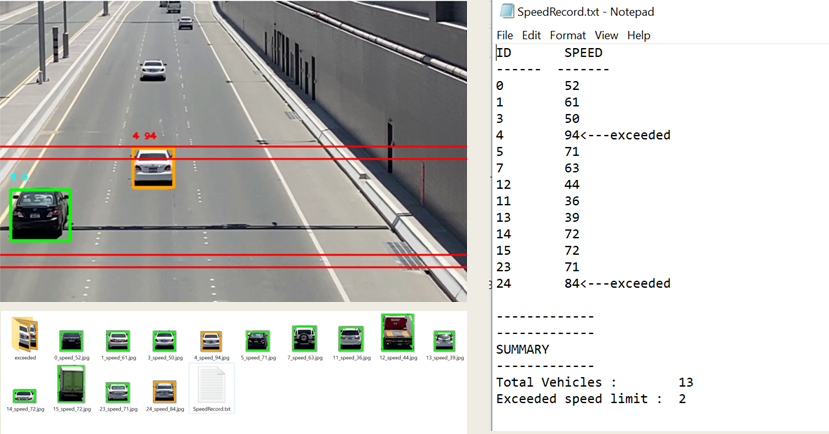


Fig 47. Speed readings

Final Output :

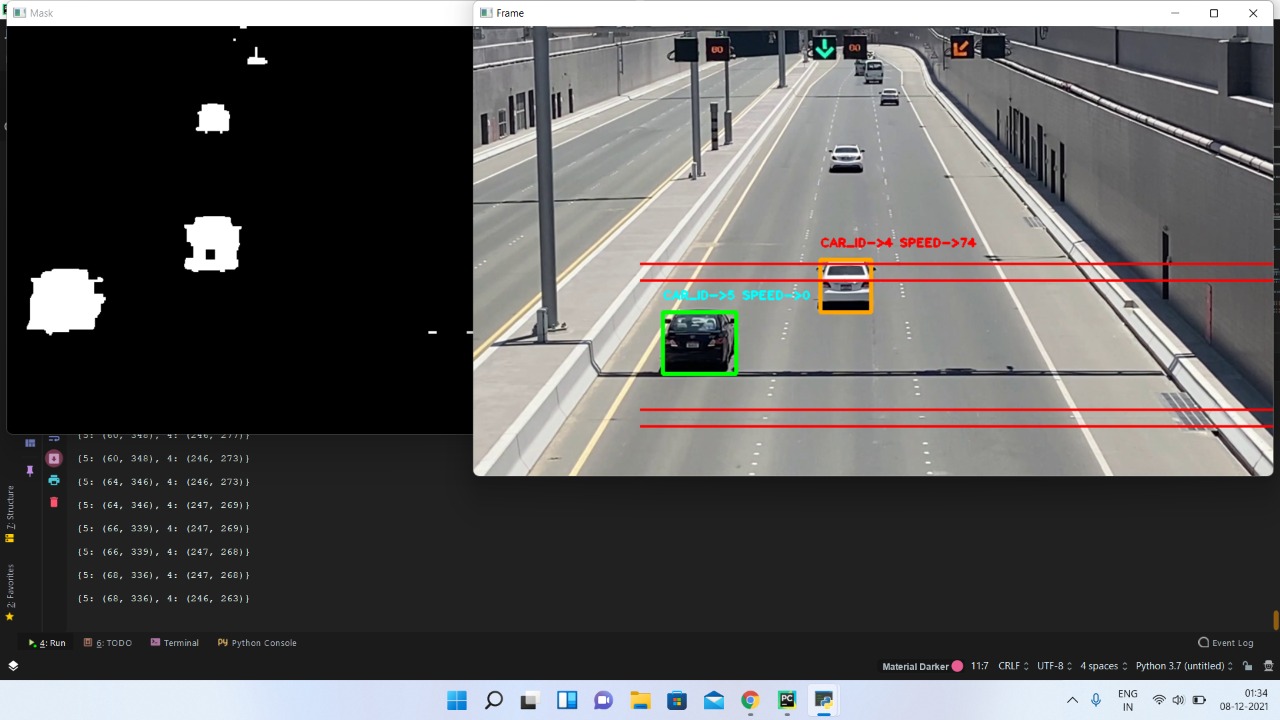


Fig 48. Simulation

**Hence the vehicle detection, tracking and speed detection is done !**

## RESULT AND DISCUSSION

An object detector has now been developed in Python version 3.9 using Open Cv library. It detects the vehicle and count the number of vehicles and also detectes the speed. The Open Cv algorithm was selected because it is an excellent computer vision library for image processing and AI, and it now incorporates GPU acceleration for synchronised operation.

The input video is served to process, the frames from the video, frames are then converted into the blobs, the pixels are then read from frames. Then object detection model is called for each blob the trained model detects the object and creates the bonding boxes for each detected object. A tracker object is made and vehicles are tracked and assigned an unique Id to achieve vehicle counting and classification. Also tracking helped us to detect the speed of vehicles.

The proposed system can be used for traffic surveillance, with cameras on traffic lights further improvements can be done on the proposed system to make it real time detection and tracking. This method has led the way for using object detection models for tracking, counting and speed detecttion. Object detection model enhancements can help increase the system's speed and accuracy.

## CONCLUSION

There are a lot of potential applications of car detection and tracking due to the development in motorways, highways, and traffic congestion. In this project we have demonstrated a vehicle detection system for effective detection and counting of vehicles and getting the speed of vehicles running on roads.

The main aim of this method is to detect the moments of vehicles by analysing camera pictures . The vehicle counting mechanism takes video from a single camera and detects and counts moving cars. Speed of the vehicle in the same video is detected. This system is developed using OpenCV library in Python.

* + 1. APPLICATION

1. It can be used in effective signaling of traffic light. It could prove to be revolutionary in traffic light management by providing less on time of red light in case of rare traffic and more time for red light in case of dense traffic. We could get an idea of the special vehicles like ambulance, which could eventually turn out to be a life savior.
2. Vehicle speed detection:- Thus, our system can enforce the traffic laws and maintain road decorum.

Helps in mountain roads:- On mountains only limited no. of vehicles are allowed and thus it can be an effective way in place of manual counting.

1. Smart city planning:- It can help in determining the appropriate roads that can hold the traffic in a place(traffic density management).
2. Surveillance:- It can be used in identifying the unwanted vehicles, thus can detect suspicious activities.

## LIMITATION

When working in this area, some limitations can arise if a low resolution camera is used, resulting in poor image quality, or if a computer with a low (GPU) graphics processing unit is employed, resulting in a slower object detection process. Also number plate of the vehicles are unreadable due to low resolution.

When two vehicles are in close proximity to one another, the algorithm may struggle to detect them effectively. In most cases, the system creates a larger bounding box and interprets it as a single vehicle. However, the created system still sends out a signal when a car is detected in such instances.

## Future Scope

Vehicle detection and counting plays an important role in an intelligent transportation system, particularly when it comes to traffic management. Several potential future improvements to the system, such as detection, tracking, counting, and classification of moving vehicles on ongoing live videos, should be conceivable. As a result, we'll be able to use it on live recordings collected right from the camera.

Another addition improvement that can be done in this project is to read the number plate of the vehicles. This can be done by using better hardwares (high resolution camera etc.) and advance image processing techniques.

## References

1. Digital image processing - 3rd ed. - R. Gonzalez, R. Woods
2. Sheeraz Memon, “A Video based Vehicle Detection, Counting and Classification System”, Mehran University of Engineering & Technology, Jamshoro, Pakistan, pp. 34-41.
3. [5] Wen-Jung Wang and Meng Ga, “Vehicle Detection And Counting In Traffic Video Based On Opencv”, Pennsylvania State University, USA, Vol 361-363, pp. 2232-2235.
4. Coursera Python Course - [https://www.coursera.org/programs/national-institute-technology-](https://www.coursera.org/programs/national-institute-technology-kurukshetra-on-coursera) [kurukshetra-on-coursera](https://www.coursera.org/programs/national-institute-technology-kurukshetra-on-coursera)
5. https://[www.who.int/news-room/fact-sheets/detail/road-traffic-injuries](http://www.who.int/news-room/fact-sheets/detail/road-traffic-injuries)
6. A. J. Kun and Z. Vamossy, "Traffic monitoring with computer vision," Proc. 7th Int. Symp. Applied Machine Intelligence and Informatics (SAMI 2009)
7. <https://www.youtube.com/watch?v=6kZftXunlTY>
8. <https://www.google.com/url?sa=t&source=web&rct=j&url=https://www.engpaper.com/vehicle-detection.html&ved=2ahUKEwiBi4n5ytH0AhU3SWwGHULyA_o4ChAWegQIGxAB&usg=AOvVaw1WWC7qx2snA1BVvxWZcSPI>
9. <https://www.google.com/url?sa=t&source=web&rct=j&url=https://dl.acm.org/doi/10.1145/3394171.3413945&ved=2ahUKEwjzos-ey9H0AhWFT2wGHVFnCpI4FBAWegQIBRAB&usg=AOvVaw3yHP4-e0-454hT-6H1q70M>
10. <https://www.google.com/url?sa=t&source=web&rct=j&url=https://www.researchgate.net/publication/326869620_Vehicle_Counting_Based_on_Vehicle_Detection_and_Tracking_from_Aerial_Videos&ved=2ahUKEwjK1Ky_ytH0AhVZSmwGHRlUCvMQFnoECBsQAQ&usg=AOvVaw0dpXbwTiUej59SvG2Y>
11. [https://www.google.com/url?sa=t&source=web&rct=jG2Y](https://www.google.com/url?sa=t&source=web&rct=j&url=https://www.researchgate.net/publication/326869620_Vehicle_Counting_Based_on_Vehicle_Detection_and_Tracking_from_Aerial_Videos&ved=2ahUKEwjK1Ky_ytH0AhVZSmwGHRlUCvMQFnoECBsQAQ&usg=AOvVaw0dpXbwTiUej59SvG2Y)

12.[https://www.google.com/url?sa=t&source=web&rct=j&url=https://www.researchgate.net/publication/326869620\_Vehicle\_Counting\_Based\_on\_Vehicle\_Detection\_and\_Tracking\_from\_Aerial\_Videos](https://www.google.com/url?sa=t&source=web&rct=j&url=https://www.researchgate.net/publication/326869620_Vehicle_Counting_Based_on_Vehicle_Detection_and_Tracking_from_Aerial_Videos&ved=2ahUKEwjK1Ky_ytH0AhVZSmwGHRlUCvMQFnoECBsQAQ&usg=AOvVaw0dpXbwTiUej59SvG2Y)

13. <https://www.youtube.com/watch?v=IdBnsNttXq0>

14. <https://www.youtube.com/watch?v=QlMsuCL_bXs>

15. [https://www.coursera.org/programs/national-institute-technology- kurukshetra-on-coursera](https://www.coursera.org/programs/national-institute-technology-%20kurukshetra-on-coursera)