

PROFESSIONAL TRAINING REPORT

at

SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY
(Deemed to be University)

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Engineering Degree in Computer Science and Engineering

by

BALIVADA HARSHITHA
(40110148)



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING

SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY
JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI –
600119, TAMILNADU

APRIL 2023



SATHYABAMA INSTITUTE OF
SCIENCE AND TECHNOLOGY (DEEMED TO BE
UNIVERSITY)

Accredited with Grade “A” by NAAC

(Established under Section 3 of UGC Act, 1956)

JEPPIAAR NAGAR, RAJIV GANDHI SALAI

CHENNAI– 600119

www.sathyabama.ac.in



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **BALIVADA HARSHITHA (Reg. No: 40110148)** who carried out the project entitled “**VEHICLE COUNTING IN TRAFFIC BASED ON IMAGE PROCESSING**” under my supervision from January 2023 to April 2023.

Internal Guide

Dr. Sankari, M.E., Ph.D.,

Head of the Department

Dr. L. Lakshmanan, M.E., Ph.D.,

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I, **BALIVADA HARSHITHA** hereby declare that the project report entitled “**VEHICLE COUNTING IN TRAFFIC BASED ON IMAGE PROCESSING**” done by me under the guidance of **Dr. SANKARI M.E., Ph.D.**, is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering Degree in Computer Science and Engineering.

DATE:

PLACE:

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to the Board **of Management** of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala M.E., Ph.D., Dean**, School of Computing, **Dr. L. Lakshmanan, M.E., Ph.,** and **Dr. S. Vigneshwari, M.E., Ph.D. Heads of the Department of Computer Science and Engineering** for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. Sankari, M.E., Ph.D.,** for her valuable guidance, suggestions and constant encouragement paved the way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

A traffic speed count based on image processing with Open CV is a system that uses computer vision techniques to estimate the speed of vehicles from a video feed captured by a camera. The system can detect and track vehicles in real-time, extract their positions and velocities, and estimate their speeds using image processing algorithms. Within this region of interest, the vehicle speed is estimated by analyzing the position changes over time. This allows the system to gather more accurate speed data for the targeted area, improving safety and reducing the likelihood of accidents

The OpenCV library provides a rich set of image processing functions, including background subtraction, object detection, and tracking algorithms. These functions can be combined to create a robust traffic speed radar system that can handle different lighting and weather conditions. The implementation of this system has several advantages, including low cost, scalability, and ease of deployment. It can be used in a variety of applications, such as traffic management, law enforcement, and transportation research. The system can also be enhanced by adding additional sensors, such as radar or lidar, to improve its accuracy and performance.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	IV
	LIST OF FIGURES	VI
1	INTRODUCTION	1
	1.1 Tools	2
	1.2 Image Processing	2
	1.3 Image Subtraction	2
2	AIM AND SCOPE	3
	2.1 Aim	5
	2.2 Scope	5
	2.3 Advantages of Proposed System	6
3	EXPERIMENTAL OR MATERIAL AND METHODS USED	7
	3.1 Imported Libraries	7
	3.2 Open CV	7
	3.3 Numpy	8
	3.4 MOG2 Algorithm	9
	3.5 Problem Statement	12
4	RESULTS AND DISCUSSION	13
	4.1 Results and Discussion	13
	4.1.1 Video Acquisition	13
	4.1.2 Masked Image	14
	4.1.3 Contour Detection	15
	4.1.4 Speed Estimation	15
	4.1.5 Create Summary	17
5	SUMMARY AND CONCLUSION	18
	REFERENCES	19
	APPENDIX	20
	A. Source code	
	B. Screenshots	

LIST OF FIGURES

FIGURE NO.	FIGURE NAME	PAGE NO.
1.2	IMAGE PROCESSING	3
1.3	IMAGE SUBTRACTION	4
3.1	IMPORTED LIBRARIES	7
3.2	OPEN CV	8
3.3	NUMPY	9
3.5.1	VECHICLE DETECTION	10
3.5.2	SPEED ESTIMATION	11
3.5.3	PROJECT MODEL BLOCK DIAGRAM	12
4.1.1	VIDEO ACQUISITION	14
4.1.2	MASKED IMAGE	15
4.1.3	CONTOUR DETECTION	15
4.1.4	SAVE VEHICLE DATA VEHICLE COUNT	16
4.1.5	SPEED RECORD	17

CHAPTER 1

INTRODUCTION

Vehicle speed and counting with OpenCV and ROI is a system that uses computer vision techniques to detect and track vehicles, estimate their speeds, and count the number of vehicles passing through a specific region of interest (ROI) in a video feed. This system is used in traffic management, transportation research, and other applications that require accurate vehicle speed and count data.

The system works by analyzing a video feed captured by a camera, detecting and tracking vehicles in the ROI using object detection and tracking algorithms. The ROI is defined as a specific area of the road where the vehicle speed and count data is required. The system then estimates the speed of each vehicle by analyzing its position changes over time and counts the number of vehicles that pass through the ROI.

The OpenCV library provides a variety of tools and algorithms for object detection, tracking, and speed estimation. These tools can be used to detect and track vehicles accurately and efficiently, estimate their speeds, and count the number of vehicles passing through the ROI.

The implementation of this system has several advantages, including accurate vehicle speed and count data, reduced manual labor, and increased efficiency. It can be used in a variety of applications, such as traffic flow analysis, congestion management, and accident prevention. By combining with additional sensors or data sources, such as GPS or weather data, the system can be enhanced to improve its accuracy and performance.

1.1 Tools:

The entire analysis was done using Python Libraries like NumPy , And OpenCV, ROI, it uses the MOG2 (Mixture of Gaussian's) algorithm and PyCharm was used.

1.2 Image Processing:

Image processing is a field of study that deals with the manipulation and analysis of digital images using computer algorithms. It involves using mathematical operations and algorithms to enhance, transform, and analyze digital images for various applications.

There are two main types of image processing: analog and digital. Analog image processing involves manipulating images using optical and mechanical devices, while digital image processing involves processing images using computer algorithms.

Digital image processing can be broadly categorized into two main areas: image enhancement and image analysis. Image enhancement involves manipulating images to improve their quality or clarity, while image analysis involves extracting information from images using algorithms.

Some common applications of image processing include image restoration, image compression, image recognition, and computer vision. Image processing is used in various fields, including medical imaging, remote sensing, video surveillance, and robotics.



Fig 1.2 IMAGE PROCESSING

1.3 Image Subtraction

Image subtraction is a process in digital image processing where one image is subtracted from another image to obtain a resultant image. The resultant image shows the difference between the two original images.

Image subtraction can be used for various purposes, such as image registration, motion detection, and image enhancement. For example, in image registration, two images of the same scene taken at different times or from different angles can be subtracted to obtain an image that shows the differences between the two images. This difference image can then be used to align the two images.

Similarly, in motion detection, two consecutive frames of a video sequence can be subtracted to obtain a difference image that highlights the regions where motion has occurred. This difference image can then be used to detect moving objects in the video sequence.

In image enhancement, image subtraction can be used to remove unwanted elements from an image.

For example, if an image contains a fixed pattern noise, a reference image of the noise pattern can be subtracted from the original image to obtain an image with reduced noise.

Overall, image subtraction is a useful technique in digital image processing that can be used for a wide range of applications.

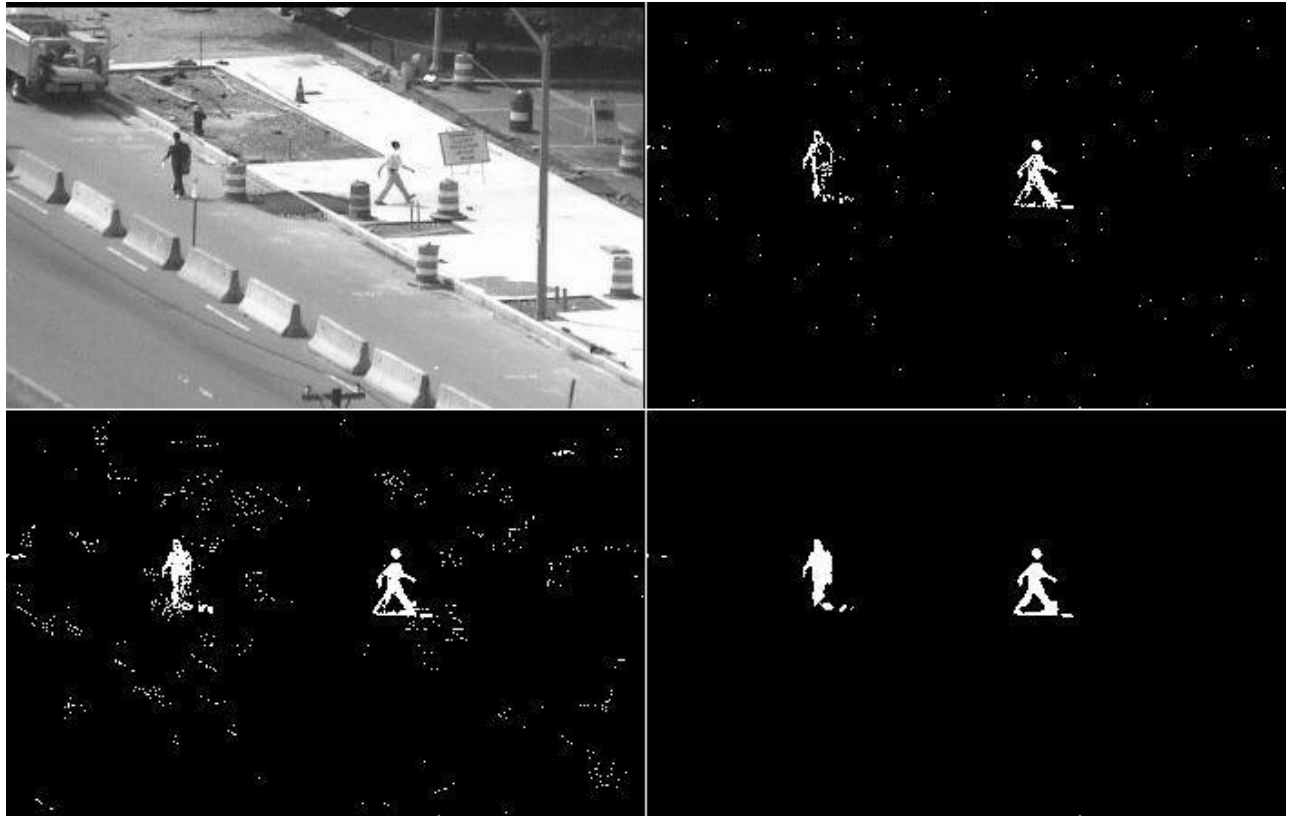


Fig.1.3 Image Subtraction

CHAPTER 2

AIM AND SCOPE

2.1 Aim:

The aim of this project is vehicle counting and speed radar is to develop a system that can accurately detect and track vehicles in a given region and estimate their speed. The system can be used for various purposes, such as traffic monitoring, surveillance, and law enforcement.

2.2 Scope:

The scope of the project would include the following:

Pre-processing: The first step would be to preprocess the video feed to enhance the quality and remove noise. This can be achieved by applying various filters and image processing techniques such as Gaussian blur, thresholding, and morphological operations.

Object detection: The next step would be to use object detection techniques such as it uses the MOG2 (Mixture of Gaussian's) algorithm for background subtraction, which is a common method for detecting moving objects in video streams.

Region of Interest (ROI) selection: Once the vehicles are detected, the system can use the ROI technique to define the region of interest and focus on a specific area where the vehicle count and speed need to be measured.

Counting: The system can count the number of vehicles passing through the defined ROI by tracking their movement and keeping a record of each vehicle that enters and exits the region.

Speed estimation: Using the tracking information and the time taken by the vehicles to pass through the ROI, the system can estimate the speed of the vehicles.

2.3 Advantages of Proposed System:

The Advantages of this Project.

- 1.Accurate data:** The system provides accurate data on vehicle counts and speed, which can be used for data-driven decision making and planning.
- 2. Cost-effective:** The system is cost-effective compared to traditional methods of traffic monitoring and surveillance, as it requires minimal hardware and can be easily implemented using open-source software.
- 3.Scalable:** The system can be easily scaled up or down to accommodate different traffic scenarios, making it suitable for different types of applications and environments.
- 4.Real-time monitoring:** The system provides real-time monitoring of traffic, allowing for quick response to changes in traffic patterns and emergencies.
- 5. User-friendly:** The system is user-friendly and easy to operate, requiring minimal training and technical expertise.
- 6. Integration:** The system can be integrated with other technologies such as GPS and GIS, allowing for more advanced analysis and decision making.

CHAPTER 3

EXPRIMENTAL OR MATERIALS AND METHODS USED

3.1 Imported Libraries:

Libraries are collections of prewritten code that users can use to optimize tasks. In project as python is used for implementation tool, it has the most libraries as compared to other programming languages. More than of 60% machine learning developers use and goes for python as it is easy to learn. As python has comparatively large collection of libraries let's look at the libraries that came in handy for mammographic dataset

Fig : 3.1 Imported Libraries



3.2 OPENCV:

OpenCV (Open Source Computer Vision) is an open-source library for computer vision and image processing. Originally developed by Intel in 1999, it is now maintained by the OpenCV community. OpenCV provides various functions and tools to help developers create computer vision applications, including image and video processing, feature detection, object detection and recognition, and machine learning.

OpenCV is written in C++ and has bindings for several programming languages, including Python, Java, and MATLAB. The Python bindings for OpenCV provide a simple and easy-to-use interface for image and video processing tasks in Python. OpenCV can be installed in Python using pip or

Anaconda package manager.

Some of the popular features of OpenCV include:

- 1) Image processing and manipulation
- 2) Object detection and tracking
- 3) Feature detection and extraction
- 4) Camera calibration and 3D reconstruction
- 5) Machine learning and deep learning algorithms for computer vision tasks
- 6) Support for multiple platforms including Windows, Linux, macOS, iOS, and Android.
- 7) OpenCV is widely used in various industries, including robotics, automotive, security, and healthcare, among others.



Fig 3.2. OPEN CV LIBRARIE

3.3 NumPy:

NumPy (short for Numerical Python) is a popular open-source library for numerical computing in Python. It provides support for multidimensional arrays, mathematical functions, linear algebra, Fourier analysis, and more. NumPy is a fundamental library for scientific computing in Python and is used in many fields, including physics, engineering, finance, data science, and machine learning.

The main object of NumPy is the `ndarray`, which is a multidimensional array of elements of the same type. The `ndarray` object provides fast and efficient operations on arrays, such as element-wise arithmetic operations, matrix multiplication, and reshaping. NumPy also provides a wide range of functions for array manipulation, including indexing, slicing, concatenation, and splitting.

Some of the popular features of NumPy include:

Fast and efficient computation with multidimensional arrays

Broadcasting, which allows for efficient element-wise operations on arrays of different shapes and sizes

Linear algebra and Fourier analysis capabilities

Tools for random number generation and statistical analysis

Integration with other scientific computing libraries, such as SciPy and Matplotlib.

NumPy can be installed in Python using pip or Anaconda package manager. Once installed, it can be imported in Python using `import numpy as np.`



Fig 3.3 NUMPY

3.4 MOG2 Algorithm:

This code uses the MOG2 (Mixture of Gaussians) algorithm for object detection. MOG2 is a background subtraction algorithm that separates foreground objects from the background in a video stream. It models the background as a mixture of several Gaussian distributions and detects any pixel that does not fit this background model as foreground.

MOG2 adapts to changes in illumination and performs well in dynamic scenes where the background can change over time. The MOG2 algorithm is implemented in OpenCV using the `cv2.createBackgroundSubtractorMOG2` function. In this code, the `object_detector` variable is created using this function to detect objects in the video stream.

3.5 PROBLEM STATEMENT:

The objective of this project is to create a traffic count and speed using Image Processing in Python by using OpenCV and TensorFlow. When it comes to tracking the speed of vehicles on a segment of road, the vital steps of this projects is:

- 1) Vehicle Detection
- 2) Speed estimation
- 3) Capturing vehicle picture
- 4) Counting vehicles

3.5.1 VEHICLE DETECTION:

As the background of the vehicles is stationary (as the speed camera is stationary) image subtraction is used to detect moving vehicle.

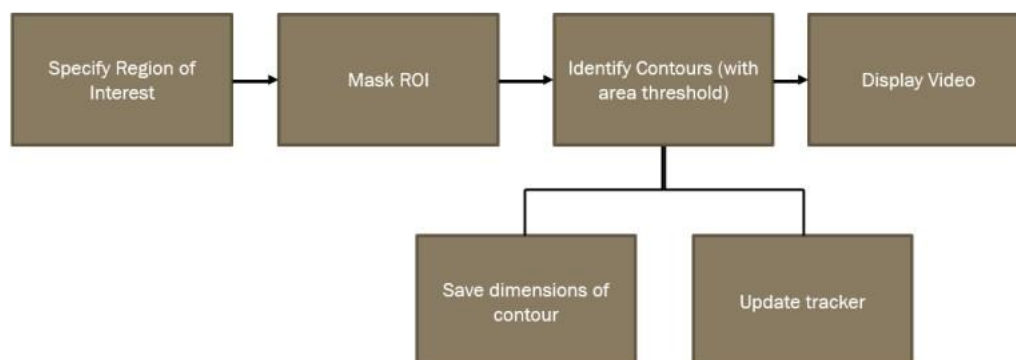
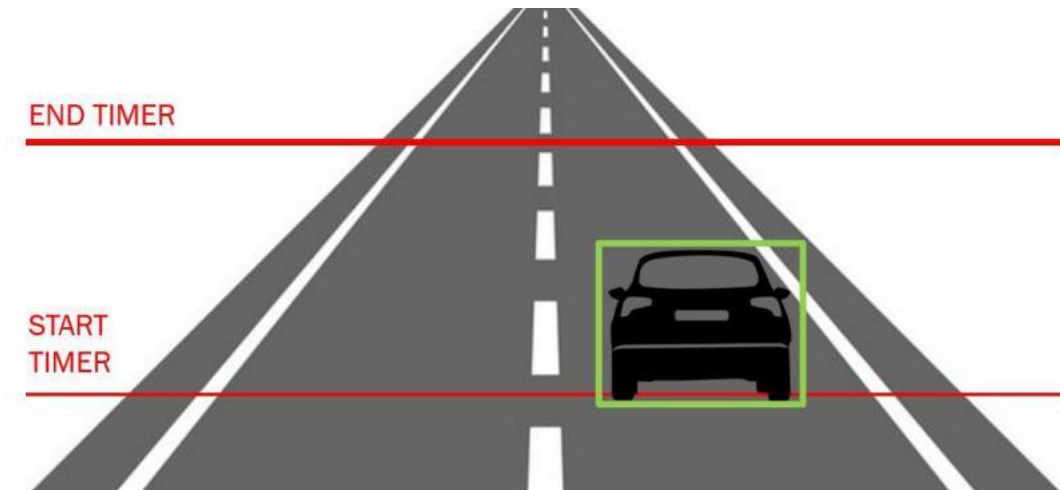


Fig 3.5. 1 VEHICLE DETECTION FLOWCHART

3.5.2 Speed Estimation :

The speed of a vehicle can be estimated when a tracked vehicle covers a segment of road



3.5.2 SPEED ESTIMATION

speed estimation is being performed by the EuclideanDistTracker class. This class is responsible for tracking objects in the video frames using the bounding boxes detected by the object detector.

To estimate the speed of the tracked objects, the EuclideanDistTracker class uses a simple distance-based method. It calculates the Euclidean distance between the centroid of the current bounding box and the centroid of the previous bounding box for the same object. Then, it divides this distance by the time elapsed between the two frames to get the speed of the object in pixels per second.

The time elapsed between the two frames is calculated by dividing the frame rate (which is set to 25fps in this code) by the number of frames that have elapsed between the two frames.

Here, `tracker.getsp(id)` gets the speed of the object with ID `id`. If the object is being tracked for the first time, `tracker.f[id]` is set to 1. `tracker.capture()` is called to capture the object's speed, along with its bounding box coordinates and ID.

3.5.3 Capturing Vehicle Pictures

Image Based on Contour detection, the image of a particular ID is saved to a folder along with the speed. Picture is saved as soon as the speed is estimated

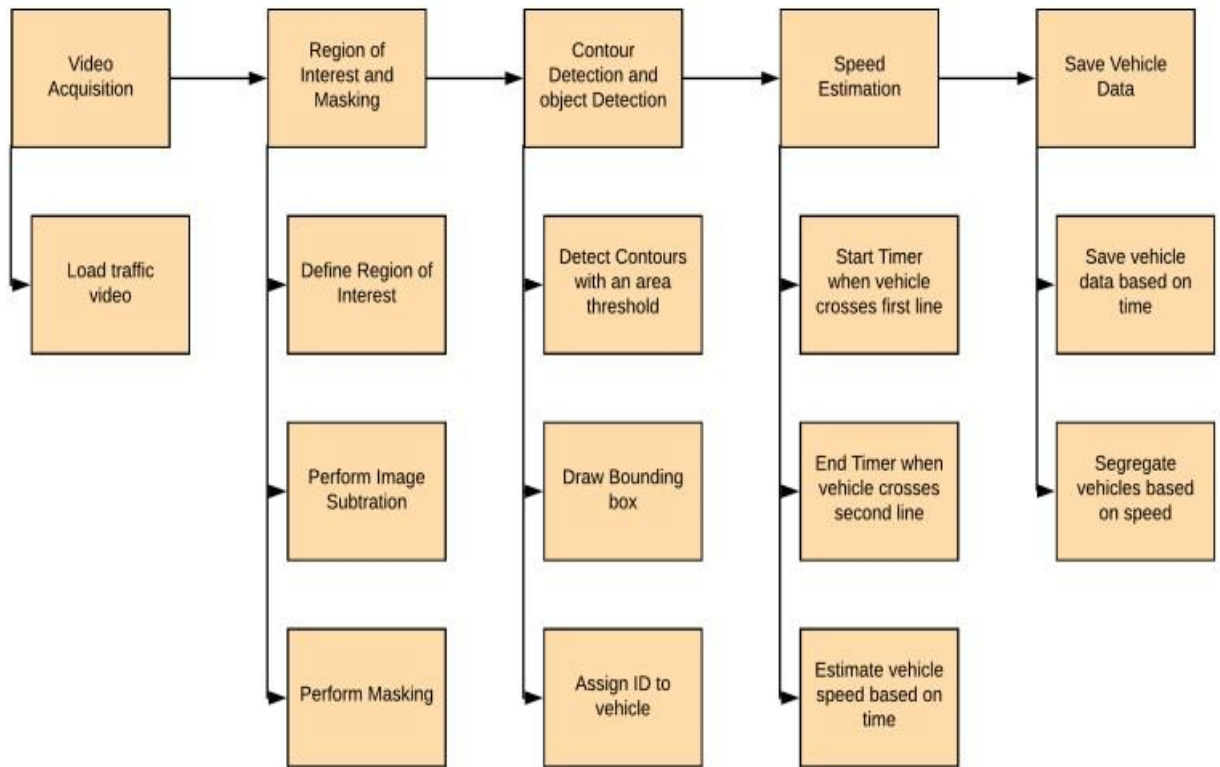


Fig 3.5.3 Project Model Block Diagram

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Results and Discussion:

Vehicle counting and speed radar systems are important tools for traffic monitoring and management. These systems use various technologies, such as cameras, radar, and sensors, to track the movement of vehicles on roads and highways. In this section, we will discuss the result and implications of using such systems.

The results of using vehicle counting and speed radar systems can provide valuable insights into traffic patterns and vehicle speeds on roads and highways. By analyzing the data collected by these systems, traffic engineers and planners can identify congested areas, optimize traffic flow, and improve road safety. For example, vehicle counting data can be used to determine the volume of traffic on a road or intersection, while speed radar data can be used to monitor vehicle speeds and identify areas where speed limits are being exceeded.

One of the main advantages of using vehicle counting and speed radar systems is their ability to provide real-time data. This data can be used to quickly identify traffic congestion or accidents and alert authorities to take appropriate action. Additionally, this data can be used to monitor traffic trends over time and inform long-term traffic planning and infrastructure development.

4.1.1 Video Acquisition

The video used in this project is a street view in Abu Dhabi. The number plates of the vehicle in the video are however not clearly visible



Fig 4.1.1 Video Acquisition

Video acquisition refers to the process of capturing digital video from a camera or other video source. There are various technologies and methods for video acquisition, depending on the type of video source and the intended application.

One common method of video acquisition is using a digital camera or a smartphone camera. These cameras typically capture video in a compressed digital format, such as MPEG or H.264, which can be easily stored and processed on a computer or other digital device.

Another method of video acquisition is using specialized hardware, such as video capture cards or USB video capture devices. These devices are typically used to capture video from analog sources, such as VHS tapes or security cameras. The captured video can then be stored and processed on a computer or other digital device

4.1.2 Masked Image

Region of Interest (ROI) takes a smaller portion of the original video. On this ROI, Image subtraction is performed to detect a moving vehicle. (Image Subtraction helps find the difference between two frames). Masking is performed to make the moving vehicles appear white and the rest of the image black



Fig 4.1.2 Masked Image

4.1.3 Contour Detection

Based on the area threshold of number of pixels, the contours are detected. The threshold is used to avoid detecting contours of smaller moving objects that are not vehicles. The object is tracked based on the distance between two contours between frames. An ID is assigned to each contour..

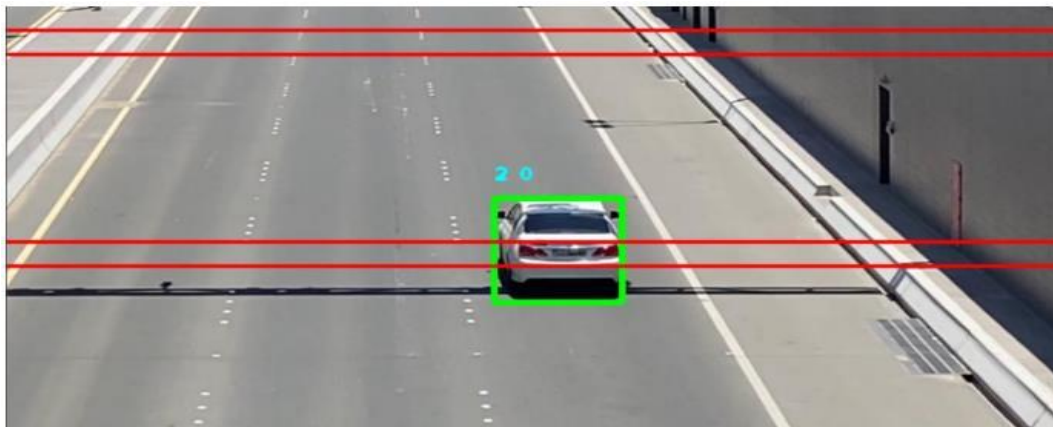
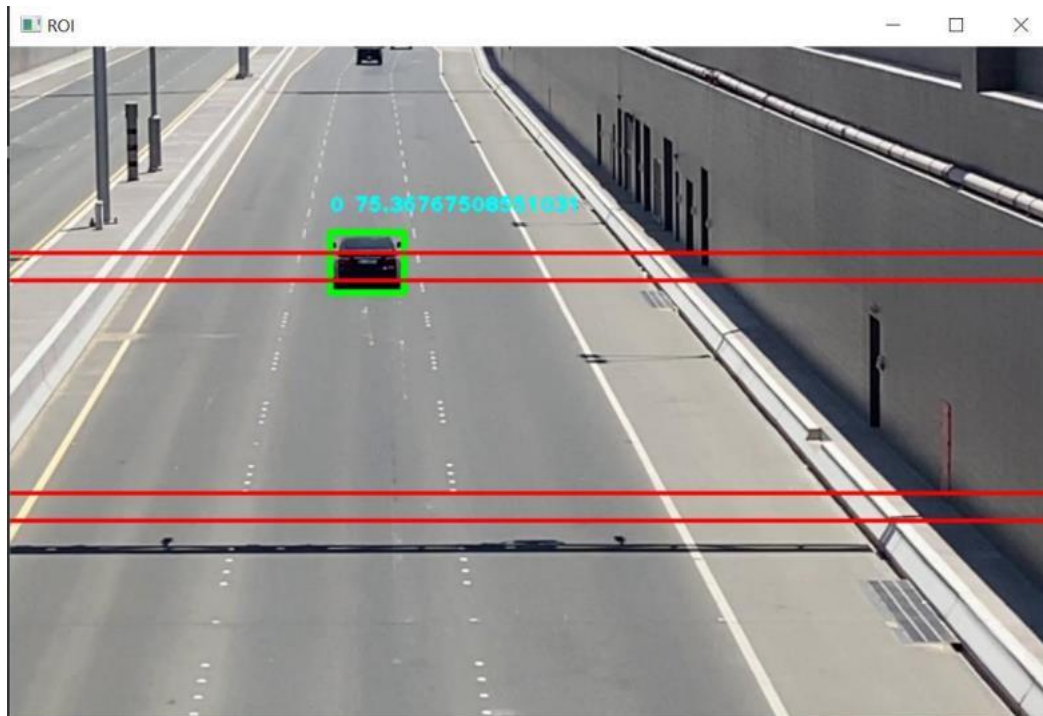


Fig 4.1.3 Contour Detection

4.1.4 Speed Estimation

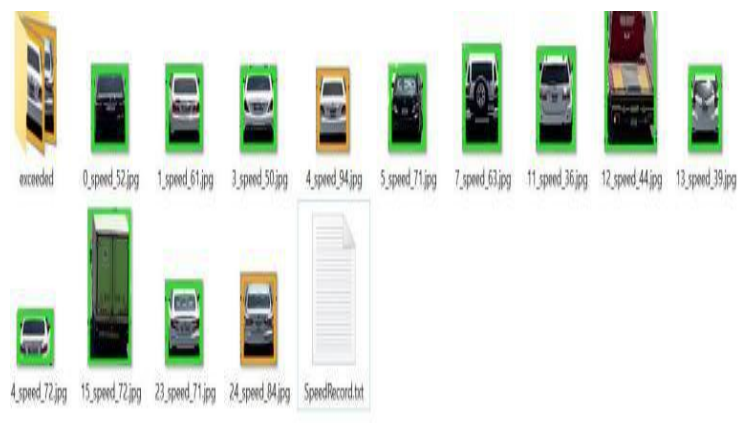
Time difference between the position of a vehicle is calculated and the speed is estimated based on a formula. The timer starts when the vehicle crosses the first line, and the timer ends when the

vehicle crosses the second line. The speed is displayed on top of the bounding box only when the vehicle crosses both the lines



4.1.4 Save Vechile Data

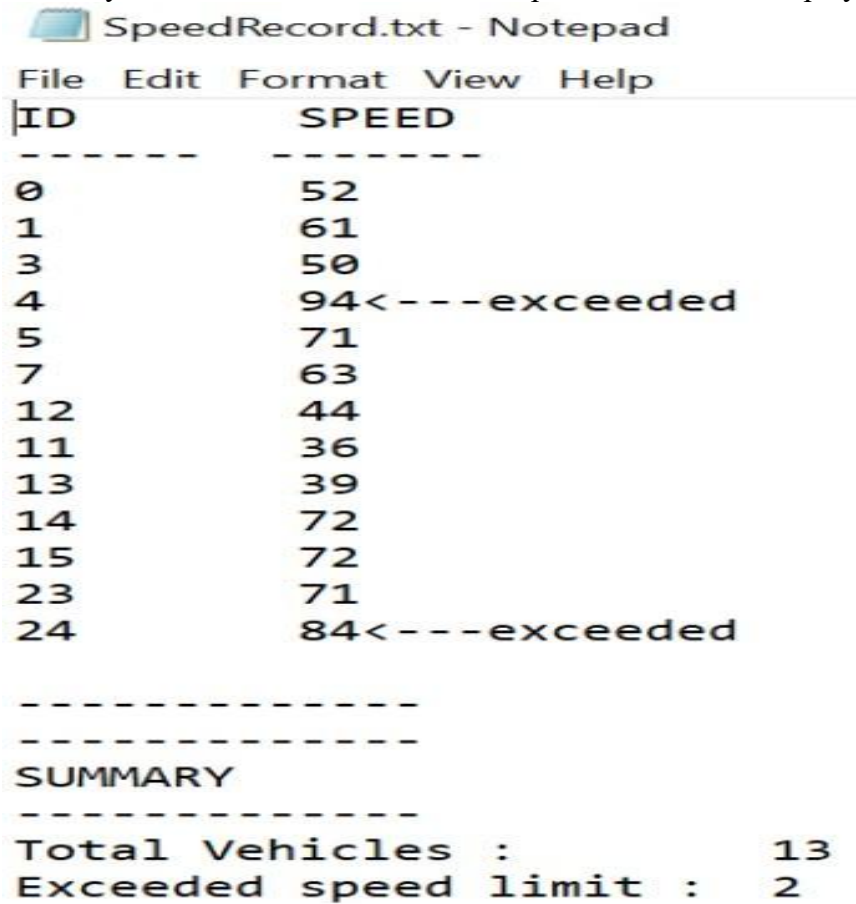
The picture of the bounding box (the vehicle) is saved into a file along with the speed. Vehicles crossing the speed limit is segregated into a separate folder.



4.1.4 VEHICLE COUNT

4.1.5.Create Summary

The vehicle data is saved in a text file. The vehicles that exceeded the speed limit are pointed. A summary of number of vehicles and the speed violators are displayed



```
File Edit Format View Help
ID      SPEED
-----
0        52
1        61
3        50
4       94<---exceeded
5        71
7        63
12       44
11       36
13       39
14       72
15       72
23       71
24      84<---exceeded

-----
SUMMARY
-----
Total Vehicles :      13
Exceeded speed limit :  2
```

Fig 4.1.5 Speed Record

CHAPTER 5

SUMMARY AND CONCLUSION

SUMMARY :

In summary, vehicle counting and radar systems are essential tools for traffic management and safety. These systems use various technologies to capture real-time data on traffic patterns and vehicle speeds, which can be analyzed to optimize traffic flow, improve road safety, and inform long-term traffic planning. While there are challenges associated with the implementation and maintenance of these systems, their benefits in terms of reducing accidents, reducing congestion, and improving driving conditions make them a valuable investment for transportation agencies and municipalities.

CONCLUSION:

In conclusion, vehicle counting and speed radar systems play a vital role in modern traffic management and safety. These systems provide real-time data that can be used to identify congested areas, optimize traffic flow, and improve road safety. By monitoring traffic patterns and vehicle speeds, transportation agencies and municipalities can make informed decisions about infrastructure development and long-term traffic planning.

The implementation of these systems is not without its challenges, such as cost and technical expertise. However, the benefits of these systems outweigh these concerns, as they can help prevent accidents, reduce congestion, and improve the overall driving experience for motorists.

In the future, advancements in technology are likely to improve the capabilities of vehicle counting and speed radar systems even further, enabling authorities to develop more effective traffic management strategies and improve road safety. Ultimately, the continued investment in these systems is crucial to creating a safer, more efficient transportation network that benefits everyone.

REFERNECES

- [1] ScienceDirect – 2020 - Determining Vehicle Speeds Using Convolutional Neural Networks by Alexander Grents, Vitalii Varkentin*, Nikolay Goryaev
- [2] ScienceDirect – 2020 - Detection of Vehicle Position and Speed using Camera Calibration and Image Projection Methods by Alexander A S Gunawana,* , Deasy Aprilia Tanjung, Fergyanto E. Gunawan
- [3] ScienceDirect – 2020 - Vehicle speed measurement model for video-based systems by Saleh Javadi , Mattias Dahl, Mats I. Pettersson [6] YouTube 25th March 2020 – OpenCV – Murtaza's Workshop
- [4] "Real-time Vehicle Detection and Counting from Traffic Video" by S. Pandey and S. Pal. This paper presents a real-time system for vehicle detection and counting from traffic video, based on a combination of background subtraction and object detection techniques. The paper includes experiments on a dataset of traffic videos.
- [5] "Vehicle Counting in Traffic Surveillance System Using Image Processing Techniques" by K. M. Basha and P. Venkata Krishna. This paper proposes a system for vehicle counting in traffic surveillance, based on image processing techniques. The system uses edge detection and morphological operations to detect and count vehicles. The paper includes experiments on a dataset of traffic images.
- [6] "Vehicle Counting and Speed Estimation in Traffic Scenes Based on Convolutional Neural Network" by X. Liu, Z. Ma, and J. Jia. This paper presents a method for vehicle counting and speed estimation in traffic scenes, based on a CNN. The method uses a sliding window approach to detect vehicles, and a Kalman filter to estimate their speed. The paper includes experiments on a dataset of traffic videos.

SOURCE CODE

```
import cv2
from tracker2 import *
import numpy as np
end = 0

#Creator Tracker Object
tracker = EuclideanDistTracker()

#cap = cv2.VideoCapture("Resources/traffic3.mp4")
cap = cv2.VideoCapture("traffic4.mp4")
f = 25
w = int(1000/(f-1))

#Object Detection
object_detector = cv2.createBackgroundSubtractorMOG2(history=None,varThreshold=None)

#100,5

#KERNALS
kernalOp = np.ones((3,3),np.uint8)
kernalOp2 = np.ones((5,5),np.uint8)
kernalCl = np.ones((11,11),np.uint8)
fgbg=cv2.createBackgroundSubtractorMOG2(detectShadows=True)
kernal_e = np.ones((5,5),np.uint8)

while True:
    ret,frame = cap.read()
    if not ret:
        break
    frame = cv2.resize(frame, None, fx=0.5, fy=0.5)
    height,width,_ = frame.shape
    #print(height,width)
    #540,960

    #Extract ROI
    roi = frame[50:540,200:960]

    #MASKING METHOD 1
    mask = object_detector.apply(roi)
    _, mask = cv2.threshold(mask, 250, 255, cv2.THRESH_BINARY)

    #DIFFERENT MASKING METHOD 2 -> This is used
    fgmask = fgbg.apply(roi)
    ret, imBin = cv2.threshold(fgmask, 200, 255, cv2.THRESH_BINARY)
    mask1 = cv2.morphologyEx(imBin, cv2.MORPH_OPEN, kernalOp) mask2
    = cv2.morphologyEx(mask1, cv2.MORPH_CLOSE, kernalCl) e_img =
    cv2.erode(mask2, kernal_e)
```

```

contours,_ = cv2.findContours(e_img,cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE)
detections = []

for cnt in contours:
    area = cv2.contourArea(cnt)
    #THRESHOLD
    if area > 1000:
        x,y,w,h = cv2.boundingRect(cnt)
        cv2.rectangle(roi,(x,y),(x+w,y+h),(0,255,0),3)
        detections.append([x,y,w,h])

#Object Tracking
boxes_ids = tracker.update(detections)
for box_id in boxes_ids:
    x,y,w,h,id = box_id

    if(tracker.getsp(id)<tracker.limit()):
        cv2.putText(roi,str(id)+" "+str(tracker.getsp(id))),(x,y-15),
cv2.FONT_HERSHEY_PLAIN,1,(255,255,0),2)
        cv2.rectangle(roi, (x, y), (x + w, y + h), (0, 255, 0), 3)
    else:
        cv2.putText(roi,str(id)+ " "+str(tracker.getsp(id))),(x, y-15),cv2.FONT_HERSHEY_PLAIN,
1,(0, 0, 255),2)
        cv2.rectangle(roi, (x, y), (x + w, y + h), (0, 165, 255), 3)

    s = tracker.getsp(id)
    if (tracker.f[id] == 1 and s != 0):
        tracker.capture(roi, x, y, h, w, s, id)

# DRAW LINES

cv2.line(roi, (0, 410), (960, 410), (0, 0, 255), 2)
cv2.line(roi, (0, 430), (960, 430), (0, 0, 255), 2)

cv2.line(roi, (0, 235), (960, 235), (0, 0, 255), 2)
cv2.line(roi, (0, 255), (960, 255), (0, 0, 255), 2)

#DISPLAY
#cv2.imshow("Mask",mask2)
#cv2.imshow("Erode", e_img)
cv2.imshow("ROI", roi)

key = cv2.waitKey(w-10)
if key==27:
    tracker.end()
    end=1
    Break

if(end!=1):
    tracker.end()

cap.release()

```

```
cv2.destroyAllWindows()
```

TRACKER.PY

```
import cv2
import math
import time
import numpy as np
import os
limit = 80 # km/hr

traffic_record_folder_name = "TrafficRecord"

if not os.path.exists(traffic_record_folder_name):
    os.makedirs(traffic_record_folder_name)
    os.makedirs(traffic_record_folder_name+"//exceeded")

speed_record_file_location = traffic_record_folder_name + "//SpeedRecord.txt"
file = open(speed_record_file_location, "w")
file.write("ID \t SPEED\n-----\t-----\n")
file.close()

class EuclideanDistTracker:
    def __init__(self):
        # Store the center positions of the objects
        self.center_points = {}

        self.id_count = 0
        # self.start = 0
        # self.stop = 0
        self.et = 0
        self.s1 = np.zeros((1, 1000))
        self.s2 = np.zeros((1, 1000)) self.s =
        = np.zeros((1, 1000)) self.f =
        np.zeros(1000) self.capf =
        np.zeros(1000) self.count = 0
        self.exceeded = 0

        def update(self, objects_rect):
            objects_bbs_ids = []

            # Get center point of new object for
            rect in objects_rect:
                x, y, w, h = rect
                cx = (x + x + w) // 2 cy =
                (y + y + h) // 2

                # CHECK IF OBJECT IS DETECTED ALREADY
                same_object_detected = False

                for id, pt in self.center_points.items():
                    dist = math.hypot(cx - pt[0], cy - pt[1])

                    if dist < 70:
                        self.center_points[id] = (cx, cy)
```

```

objects_bbs_ids.append([x, y, w, h, id])
same_object_detected = True

# START TIMER
if (y >= 410 and y <= 430):
    self.s1[0, id] = time.time()

# STOP TIMER and FIND DIFFERENCE
if (y >= 235 and y <= 255):
    self.s2[0, id] = time.time()
    self.s[0, id] = self.s2[0, id] - self.s1[0, id]

# CAPTURE FLAG
if (y < 235):
    self.f[id] = 1

# NEW OBJECT DETECTION
if same_object_detected is False:
    self.center_points[self.id_count] = (cx, cy)
    objects_bbs_ids.append([x, y, w, h, self.id_count])
    self.id_count += 1
    self.s[0, self.id_count] = 0
    self.s1[0, self.id_count] = 0
    self.s2[0, self.id_count] = 0

# ASSIGN NEW ID to OBJECT
new_center_points = {}
for obj_bb_id in objects_bbs_ids:
    _, _, _, _, object_id = obj_bb_id
    center = self.center_points[object_id]
    new_center_points[object_id] = center

self.center_points = new_center_points.copy()
return objects_bbs_ids

# SPEED FUNCTION
def getsp(self, id):
    if (self.s[0, id] != 0):
        s = 214.15 / self.s[0, id]
    else:
        s = 0

    return int(s)

# SAVE VEHICLE DATA
def capture(self, img, x, y, h, w, sp, id):
    if (self.capf[id] == 0):
        self.capf[id] = 1
        self.f[id] = 0
        crop_img = img[y - 5:y + h + 5, x - 5:x + w + 5]
        n = str(id) + "_speed_" + str(sp)
        file = traffic_record_folder_name + '/' + n + '.jpg'
        cv2.imwrite(file, crop_img)
        self.count += 1
        file = open(speed_record_file_location, "a")
        if (sp > limit):

```

```

        file2 = traffic_record_folder_name + '//exceeded//' + n + '.jpg'
        cv2.imwrite(file2, crop_img)
        file.write(str(id) + " \t " + str(sp) + "<---exceeded\n")
        self.exceeded += 1 else:
        file.write(str(id) + " \t " + str(sp) + "\n")
    file.close()

# SPEED_LIMIT
def limit(self):
    return limit

# TEXT FILE SUMMARY
def end(self):
    file = open(speed_record_file_location, "a")
    file.write("\n-----\n")
    file.write("-----\n")
    file.write("SUMMARY\n")
    file.write("-----\n")
    file.write("Total Vehicles : \t" + str(self.count) + "\n")
    file.write("Exceeded speed limit : \t" + str(self.exceeded))
    file.close()

```

Video Acquisition:

```
cap = cv2.VideoCapture("Resources/traffic3.mp4")
```

Region of Interest and Making

```

#KERNALS
kernalOp = np.ones((3,3),np.uint8)
kernalOp2 = np.ones((5,5),np.uint8)
kernalCl = np.ones((11,11),np.uint8)
fgbg=cv2.createBackgroundSubtractorMOG2(detectShadows=True)

#MASKING
fgmask = fgbg.apply(roi)
ret, imBin = cv2.threshold(fgmask, 200, 255, cv2.THRESH_BINARY)
mask1 = cv2.morphologyEx(imBin, cv2.MORPH_OPEN, kernalOp)
mask2 = cv2.morphologyEx(mask1, cv2.MORPH_CLOSE, kernalCl)

```

Contour Detection

```

contours, _ = cv2.findContours(mask2, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
detections = []

for cnt in contours:
    #print(cnt)
    area = cv2.contourArea(cnt)
    if area > 1000:
        x,y,w,h = cv2.boundingRect(cnt)
        cv2.rectangle(roi, (x,y), (x+w,y+h), (0,255,0), 3)
        detections.append([x,y,w,h])

```

Object Tracking


```

for rect in objects_rect:
    x, y, w, h = rect
    cx = (x + x + w) // 2
    cy = (y + y + h) // 2

    # Find out if that object was detected already
    same_object_detected = False

    for id, pt in self.center_points.items():
        dist = math.hypot(cx - pt[0], cy - pt[1])

        if dist < 150:
            self.center_points[id] = (cx, cy)
            #print(self.center_points)
            objects_bbs_ids.append([x, y, w, h, id])
            same_object_detected = True

```

Speed Estimation

8.5.1 Timer Start and Stop

```

if (y >= 325 and y <= 345):
    self.s1[id] = time.time()

if (y >= 150 and y <= 170):
    self.s2[id] = time.time()

    self.s[id] = self.s2[id] - self.s1[id]

```

Speed Formula

```

def getsp(self,id):
    if (self.s[id]!=0):
        s = 439.8/self.s[id]
    else:
        s = 0
    return s

```

Drawing Rectangles and displaying on the screen

```

for box_id in boxes_ids:
    x,y,w,h,id = box_id

    cv2.putText(roi,str(id)+" "+str(tracker.getsp(id)),(x,y-15),
cv2.FONT_HERSHEY_PLAIN,1,(255,255,0),2)
    #print(tracker.getsp(id))
    cv2.rectangle(roi, (x, y), (x + w, y + h), (0, 255, 0), 3)

```

Drawing Reference Lines

```

cv2.line(roi, (0, 325), (960, 325), (0, 0, 255), 2)
cv2.line(roi, (0, 345), (960, 345), (0, 0, 255), 2)

cv2.line(roi, (0, 150), (960, 150), (0, 0, 255), 2)
cv2.line(roi, (0, 170), (960, 170), (0, 0, 255), 2)

```

Save Vehicle Image and Speeds

```

def capture(self, img, x, y, h, w, sp, id):
    if(self.capf[id]==0):
        self.capf[id] = 1
        self.f[id]=0
        crop_img = img[y-5:y + h+5, x-5:x + w+5]
        n = str(id)+"_speed_"+str(sp)
        file = 'D://TrafficRecord//' + n + '.jpg'
        cv2.imwrite(file, crop_img)
        self.count += 1
        file1 = open("D://TrafficRecord//SpeedRecord.txt", "a")
        if(sp>limit):
            file2 = 'D://TrafficRecord//exceeded//' + n + '.jpg'
            cv2.imwrite(file2, crop_img)
            file1.write(str(id)+" \t "+str(sp)+"<---exceeded\n")
            self.exceeded+=1
        else:
            file1.write(str(id) + " \t " + str(sp) + "\n")
        file1.close()

```

Create Summary

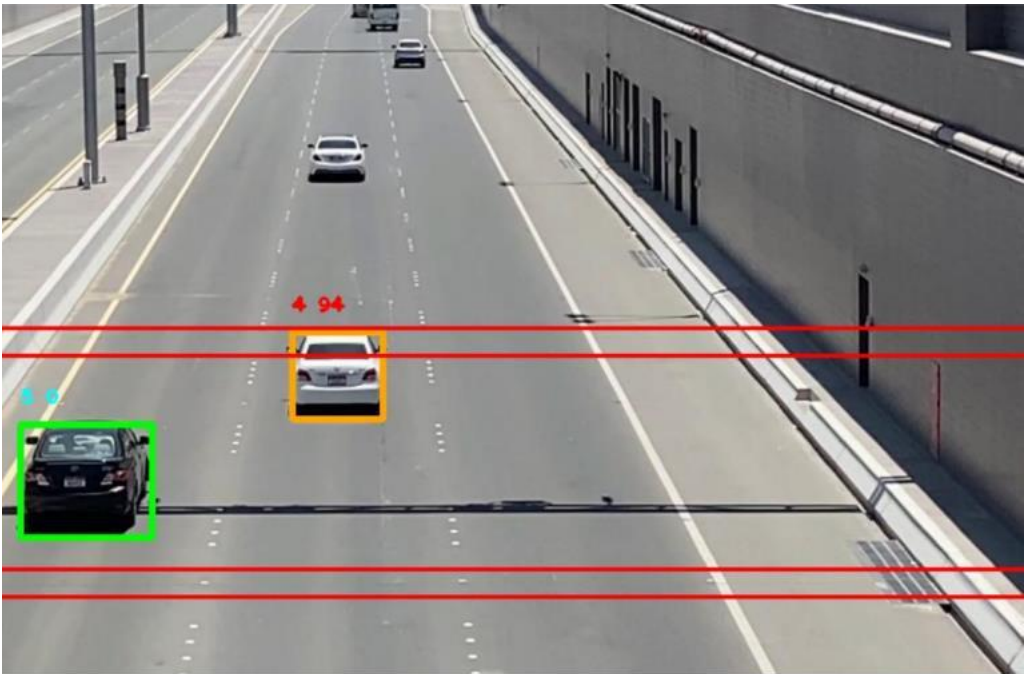
```

def end(self):
    file = open("D://TrafficRecord//SpeedRecord.txt", "a")
    file.write("\n-----\n")
    file.write("-----\n")
    file.write("SUMMARY\n")
    file.write("-----\n")
    file.write("Total Vehicles :\t"+str(self.count)+"\n")
    file.write("Exceeded speed limit :\t"+str(self.exceeded))
    file.close()

```

SCREENSHOTS

Speed Detection



Saved Image Pictures



Saved Vehicle Data

SpeedRecord.txt - Notepad

File Edit Format View Help

ID	SPEED
0	52
1	61
3	50
4	94<---exceeded
5	71
7	63
12	44
11	36
13	39
14	72
15	72
23	71
24	84<---exceeded

SUMMARY

Total Vehicles : 13
Exceeded speed limit : 2