

**MADHAV INSTITUTE OF TECHNOLOGY AND  
SCIENCE, GWALIOR**

**DEPARTMENT OF COMPUTER SCIENCE &  
ENGINEERING**



**MINOR PROJECT REPORT**

**ON TITLE**

Implementation and analysis of automated vehicle  
speed detector using deep learning approach

**MENTORED BY:**

Prof. Smita Parte

**SUBMITTED BY:**

Md. Asif Farhan (CS201070)

Mohit Tripathi (CS201071)

## **CERTIFICATE**

This is to certify that the project is titled “Implementation and analysis of automated vehicle speed detector using deep learning approach”.

This project is submitted by MOHD ASIF FARHAN (0901CS201070) and MOHIT TRIPATHI (0901CS201071) of Madhav Institute of Technology and Science in fulfilment of the requirements for BTech. in Computer Science and Engineering. This project was an authentic work done by them under my supervision and guidance. This project has not been submitted to any other institution for the award of an Under Graduation.

Date: - 17/11/22

Signature of Supervisor

# **INDEX**

1. Abstract
2. Introduction
3. Objective
4. Hardware & Software Used
5. Flow Chart
6. Results
7. Conclusion

## **ABSTRACT**

As the technology is growing immensely in recent years, certain improvisations need to be made. People are used to drive their vehicles above the prescribed speed limits. So, there is a need at the administration side to develop an automated system to detect the speed of vehicles and impose penalty if found guilty. We will develop an automated system that can help the administration in detection of speeds of vehicles without human interference.

We will be using OpenCV library for implementing our project. OpenCV is a huge open-source library for computer vision, machine learning, and image processing. OpenCV supports wide variety of programming languages like Python, C++, Java etc. It can even process images and videos to identify objects, faces or even the handwriting of a human.

In this project we will be using a recorded video to detect the speed of vehicles. We will first detect the vehicle movements and then analyse these movements to detect the speed of vehicles. We will also supply smoothing techniques to make it more usable. We can detect the speed of vehicles without human interference hence the name automated vehicle speed detector.

# **INTRODUCTION**

- ➔ Vehicle speed measurement model for video-based systems: -
  - This paper uses segmentation of the road into strips for estimation of the speed of the vehicle. If we know the length of a segment of road and the time taken to cover it, we can determine the speed. Frame-rate can be used to assess the accuracy of speed calculation.
  - A minimum speed of object is set up so as to not record unnecessary objects like bicycles and bird shadows.
- ➔ Determining vehicle speed based on video image subtraction: -
  - This paper uses deep learning to identify vehicles based on image subtraction and contour detection.

## **OBJECTIVE**

The objective of this project is to create a traffic radar 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 project are:

- Vehicle Detection
- Speed estimation
- Capturing vehicle image

**Video Acquisition:** - A video with good clarity and good fps (30-60) would be taken to record vehicles passing by on a road.

**Object Recognition:** - Deep learning would be used to identify vehicles with TensorFlow using faster RCNN or YOLO. Multiple vehicles are to be detected at a time on a road segment. Classification of vehicles into cars and trucks are also to be done.

**Speed Estimation:** - This would be done by using time taken to cover a segment of road taking into consideration the accuracy (which is mostly determined by frame-rate). The road shall be divided to multiple segments to estimate speed.

**Save Vehicle Image:** - This can be done after saving still images of the violators. Each vehicle is given a specific ID. The ID and speed details are saved to a text file. A good quality video is required to capture the number-plate of vehicles.

# **HARDWARE & SOFTWARE USED**

## **Hardware Requirements:**

### **➤ Development end:**

- System : i3
- Hard Disk : 500 GB.
- Monitor : 15" LED
- Input Devices : Keyboard, Mouse
- Ram : 2 GB

### **➤ Deployment end:**

- System : i3
- Hard Disk : 500 GB.
- Monitor : 15" LED
- Input Devices : Keyboard, Mouse
- Ram : 2 GB

## **Software Requirements:**

### **➤ Development end:**

- Operating system : Windows 7 and Upper Version
- Coding Language : Python 3.0
- Tool : PyCharm, Notepad
- Database : No Database Required

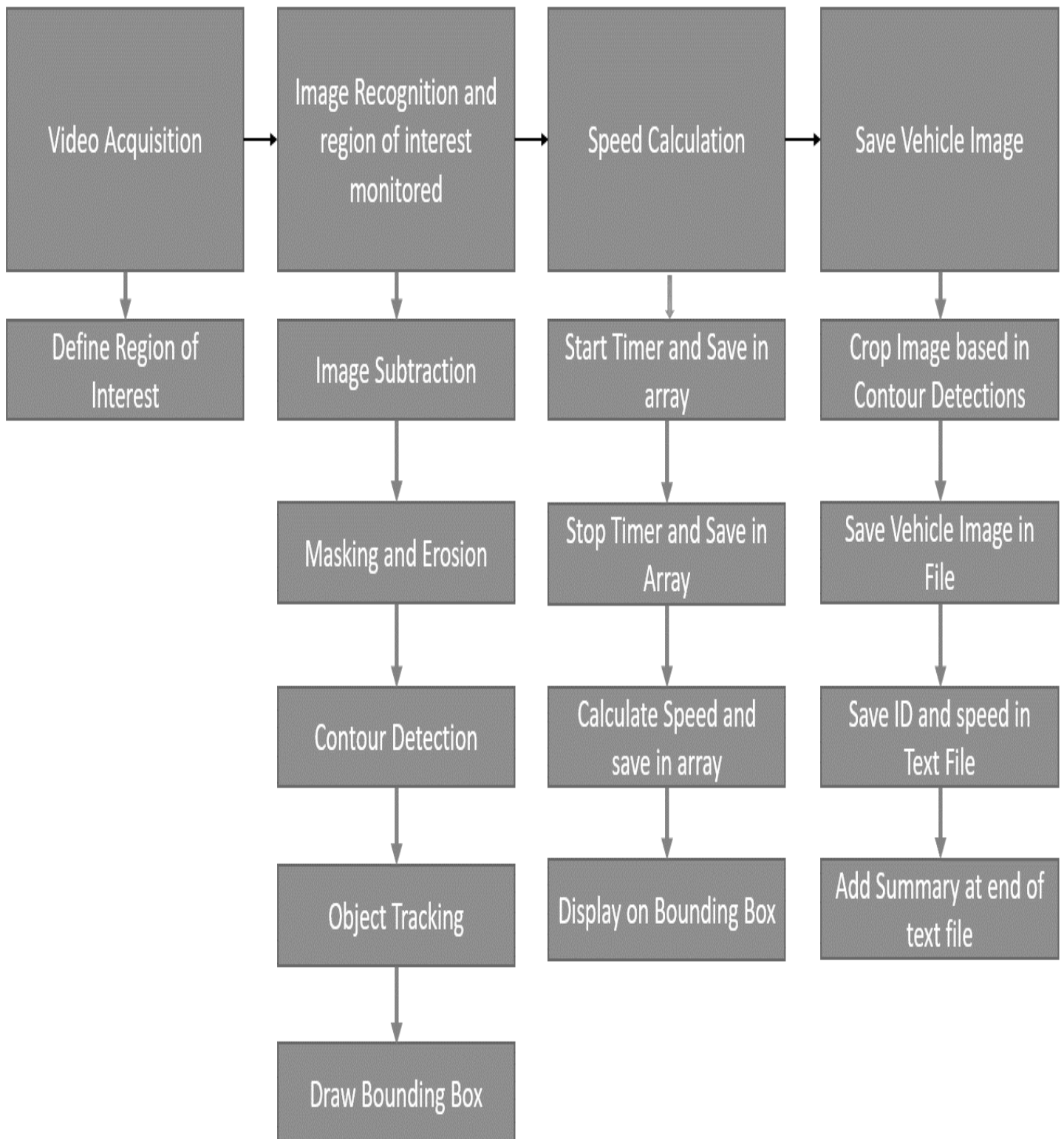
### **➤ Deployment end:**

- Operating system : Windows 7 and Upper Version.
- Coding Language : Python 3.0
- Tool : PyCharm, Notepad
- Database : No Database Required

### **➤ Technology Used:**

- Computer Vision

# **FLOW CHART**





# METHODOLOGY

## Video Acquisition: -

```
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("C:/Users/Mohit Tripathi/Desktop/Traffic.webm")
f = 25
w = int(1000/(f-1))
print(w)
```

## Object Recognition:-

```
#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()
```

```

frame = cv2.resize(frame, None, fx=1, fy=2)
height,width,_ = frame.shape
#print(height,width)
#540,960

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

#MASKING METHOD
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])

```

## **Speed Estimation:-**

### **Part 1:-**

## #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()
```

## **Part 2:-**

```
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 = 75 / self.s[0, id]
```

```
    else:
```

```
        s = 0
```

```
return int(s)
```

## Saving Vehicle Image:-

```
import cv2
import math
import time
import numpy as np

limit = 120 #km/hr

file = open("C:/Users/Mohit Tripathi/Desktop/Minor Project/SpeedRecord.txt","w")
file.write("ID \t SPEED\n-----\t-----\n")
file.close()

#SAVE VEHICLE DATA
def capture(self,img,x,y,h,w,sp,id):
    if(self.capf[id]==0):
        self.capf[id] = 1
        self.ff[id]=0
        crop_img = img[y+5:y + h+5, x+5:x + w+5]
        n = str(id)+"_speed_"+str(sp)
        file = 'C:/Users/Mohit Tripathi/Desktop/Minor Project/' + n + '.jpg'
        cv2.imwrite(file, crop_img)
        self.count += 1
        file2 = open("C:/Users/Mohit Tripathi/Desktop/Minor Project/SpeedRecord.txt", "a")
        if(sp>limit):
            file2 = 'C:/Users/Mohit Tripathi/Desktop/Minor Project/exceeded/' + n + '.jpg'
            cv2.imwrite(file2, crop_img)
            file2.write(str(id)+" \t "+str(sp)+"<---exceeded\n")
            self.exceeded+=1
        else:
            file2.write(str(id) + " \t " + str(sp) + "\n")
        file2.close()
```

```
#SPEED_LIMIT

def limit(self):
    return limit

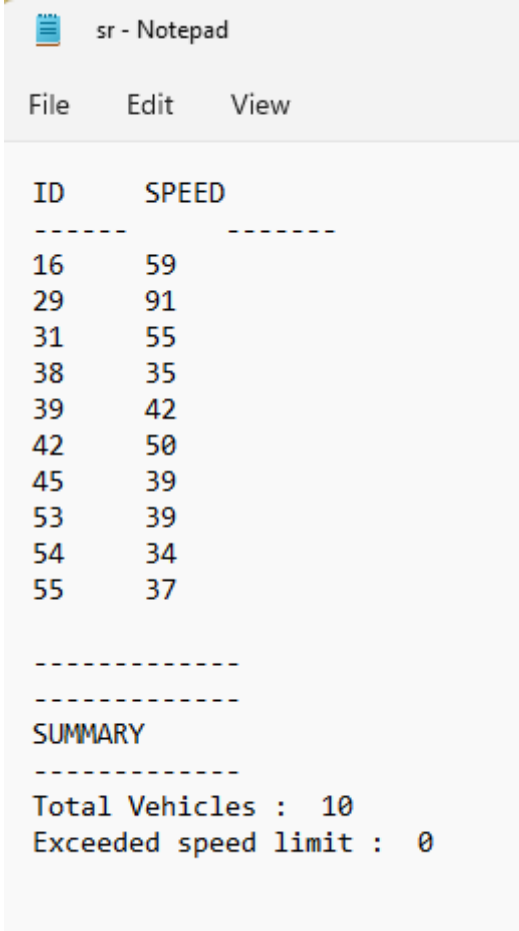
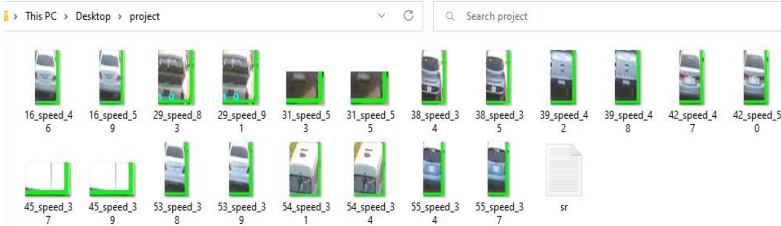
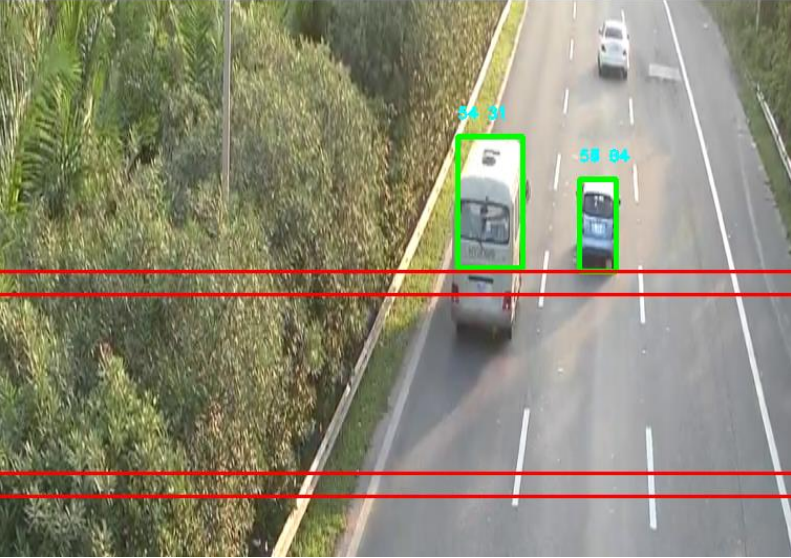
#TEXT FILE SUMMARY

def end(self):
    file = open("C:/Users/Mohit Tripathi/Desktop/Minor Project/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()
```



# RESULTS

## SAVING VEHICLE DATA: -



ID	SPEED
16	59
29	91
31	55
38	35
39	42
42	50
45	39
53	39
54	34
55	37

SUMMARY

Total Vehicles : 10  
Exceeded speed limit : 0

Under the domain of this project, first of all we have acquired a video with good clarity and good fps (30-60) to record vehicles passing by on a road.

Afterwards, we have used deep learning to identify vehicles with TensorFlow using faster RCNN.

Later we have made the program to estimate the speed by using time taken to cover a segment of road taking into consideration the accuracy (which is mostly determined by frame-rate).

Finally, we have saved the vehicle speeds and their images.

## **CONCLUSION**

Road safety and reducing accidents is a very crucial issue and must be considered at utmost priority. One must abide the rules of maintaining appropriate speed guidelines. Technological tools and tracking devices which help in monitoring the motion and speed of vehicles can help reduce the number of accidents on roads as well as trace the origins of the mishap. In this report, we have discussed the challenges and obstacles faced while implementing a system which detects a vehicle and monitors its speed and motion. We have studied methods available for object detection in video, speed estimation, and saving vehicle image. Although, some of the algorithms have been used in object detection and speed estimation gives good results, but still no algorithm can resolve all the challenges and difficulties faced in vehicle speed detection in today's generation. In the literature survey there are many sophisticated methods explained for object detection and speed estimation.

The separation of foreground and background objects and commonly preferred approaches to solve this issue. In addition, to this we have also suggested a possible formulation which can be used to detect the motion of vehicle. Furthermore, the report also talks about the speed tracking algorithm and tried to elucidate the working of these algorithms and mathematics involved behind it. Several nations are already using such systems to detect the speed and direction of vehicle. Moreover, some systems have advanced to the capacity of detecting the number plates of vehicles which are blurred for normal cameras and uses image processing algorithms to sharpen the image and extract the number plate which makes it even easier to locate the vehicle. Also, the speed breakers can be designed in such a way that they only rise when the vehicles speed is above the permissible limit. We have used OpenCV and for object detection. We have implemented an optimum solution for speed tracking and vehicle detection.

## **REFERENCES**

1. Asaad AMA, Syed IA (2009). "Object identification in video images using morphological background estimation scheme" Chapter, 22: 279-288.
2. Bailo G, Bariani M, Ijas P, Raggio M (2005). "Background estimation with Gaussian distribution for image segmentation, a fast approach," Proc. IEEE Intl. workshop on Measurement Systems for Homeland Security, Contraband Detection and Personal safety, pp. 2-5.
3. Ferrier NJ, Rowe SM, Blake A (1994). Real--time traffic monitoring. In WACV94, pp. 81—88.
4. Fumio Y, Wen L, Thuy TV (2008). "Vehicle Extraction And Speed Detection From Digital Aerial Images" IEEE International Geosciences and Remote Sensing Symposium, pp. 1134-1137.
5. Huei-Yung L, Kun-Jhih L (2004). "Motion Blur Removal and Its Application to Vehicle Speed Detection", The IEEE International Conference on Image Processing (ICIP 2004), pp. 3407-3410.