A Progress Report

on

Vehicle Speed Estimation using Object Detection and Tracking

carried out as part of the course CSE CS3270 Submitted by

Shreyas Gupta

199301260

VI-CSE

in partial fulfilment for the award of the degree

of

BACHELOR OF TECHNOLOGY

In

Computer Science & Engineering

MANIPAL UNIVERSITY JAIPUR

INSPIRED BY LIFE

Department of Computer Science & Engineering,
School of Computing and IT,
Manipal University Jaipur,
June 2022

Vehicle Speed Detection using Object Detection and Tracking

Minor Project Report

Shreyas Gupta[MU - Jaipur]
199301260

Abstract

With the ever rapidly developing world, the need for faster, better, and efficient technology is also there. As we know that Computer Vision has come a long way from what it used to be in the early 1960s. The project ‘Vehicle Speed Detection using Object Detection and Tracking’ makes use of Computer Vision, more specifically Object Detection and Tracking to estimate the speed of moving vehicles on roads and store it in a file with the timestamp associated with it.

The need for this kind of project arises as we all have seen in our apartment complexes or housing neighborhoods where ignorant drivers disregard safety and zoom by, going *way* too fast. We feel almost powerless. These drivers disregard speed limits, crosswalk areas, school zones, and ‘children at play’ signs altogether. Thus, the aim of the project is to make use of technology to catch this sort of behavior using CCTV feed.

The project is entirely built using Python version 3.7.2 as it robust and one of the preferred languages when dealing with problems concerned to the domain of Computer Vision or Machine Learning. Another reason for using Python is that for object detection and tracking we are making use of a library OpenCV which is easier to work with in Python.

CERTIFICATE

This is to certify that the project entitled "Vehicle Speed Detection using Object Detection and Tracking" is a bona fide work carried out as part of the course Minor Project : CS3270, under my guidance(Dr. Sandeep Chaurasia) by Shreyas Gupta, student of Bachelor Of Technology (B.Tech.) in Computer Science and Engineering (CSE) at the Department of Computer Science & Engineering , Manipal University Jaipur, during the academic semester VI of year 2021-22.

Place: Manipal University Jaipur

Date: 23-May-2022 Signature of the Instructor (s)

DECLARATION

I hereby declare that the project entitled “Vehicle Speed Detection using Object Detection and Tracking” submitted as part of the partial course requirements for the course Minor project (CS3270) for the award of the degree of Bachelor of Technology in Computer Science and Engineering at Manipal University Jaipur in the semester during academic year 2021-22, has been carried out by me. I declare that the project has not formed the basis for the award of any degree, associate ship, fellowship, or any other similar titles elsewhere.

Further, I declare that I will not share, re-submit, or publish the code, idea, framework and/or any publication that may arise out of this work for academic or profit purposes without obtaining the prior written consent of the Course Faculty Mentor and Course Instructor.

Signature of the Student: Shreyas Gupta

Place: Manipal University Jaipur

Date: 23-May-2022

Table of Contents

List of Figures

1. Introduction

Computer Vision, a field that wasn't known to a vast group of people just few years ago is now something that everyone knows about. The application of Computer Vision is seen in many areas from medicine to security and from traffic flow analysis to self-driving cars.

This particular project makes use of Computer Vision to estimate the speed of vehicles and log their details into a csv file if it crosses a certain threshold value of speed. Further details about the project are discussed below.

1.1 Scope of the Work

This project is being built with the idea of it being implemented on a smaller scale rather than a full scale city-wide or state-wide deployment. It can be very helpful in identifying miscreants who violate traffic rules on a whim. But if we talk about the prospects for the project, it can be implemented on a city-wide basis especially dedicated to areas where it is hard for law-enforcement agencies to keep a constant watch.

1.2 Product Scenarios

The user of this project can be anyone who is facing problems in their localities where there are people who show no regard to the traffic rules and violates them whenever they desire. We can understand the product scenario better with a case study which is discussed below.

'Mr. Sharma lives in Ambedkar Nagar colony with his family. Since there is no playground in the colony many a times his son plays with his friends on the road right outside his home. The colony has already made this clear that no vehicles should cross a speed limit of 20kmph within the colony to ensure safety of the children, but it has been observed that there are people who are not following this rule and endangering the well-being of the children. So Mr. Sharma wants a system to be developed to which is capable enough to identify such people so that they can be reported to the local authorities. Mr. Sharma isn't very tech savvy, but he wants the system to be low cost and efficient.'

2 Requirement Analysis

2.1 Functional Requirements

- Reporting Requirements – These are the information and data that an individual must supply to government agencies in case of any ill practices or law violation is found.
- Historical Data Management – This includes how the log files will be handled which includes the data of the vehicles passing by along with the time stamps.

2.2 Non-Functional Requirements

- Compatibility – Throughout the development of the project it must be made sure that the minimum hardware requirements and software versions are compatible.
- Maintainability – With changing surveillance environment how easy it is to maintain the system.
- Usability – The system should be easy for people to interact with as it might not be handled by people who are tech savvy.
- Performance – The performance aspect of the system is of utmost importance as the system would be deployed as a real-time solution and the response time and accuracy for detection and tracking should be ideal.

2.3 Use Case Scenarios

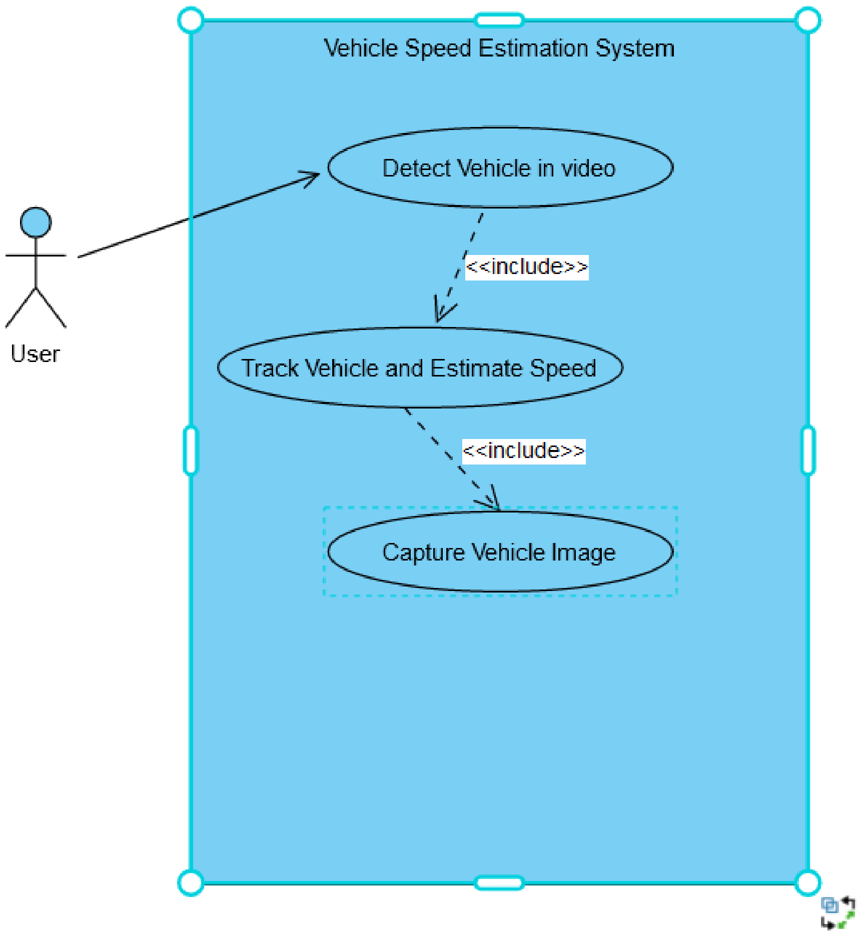


Fig 1. Use Case Diagram

- User interacts with the system through a video feed.
- The system first detects the cars in the video and the once it is detected it proceeds to tracking the same. Using two reference points the speed of the vehicle is estimated and a picture of the same is also captured for future reference.

2.4 Other Software Development Methodologies

- Incremental Model – The model that is being followed for the development of this project is an incremental model. With each new increment a new functionality is added to the project thereby the project will be completed in the stipulated time. Along with this the main reason for choosing this model is that the technology being used in the project is somewhat unfamiliar, so incremental model is best choice model for this project.

2.5 RELEVANCE

- Speeding Cameras are present in many highways in order to catch speeding vehicles and improve the safety on roads. Developed cities have more speed cameras and have more well monitored roads. Cameras today not only catch speed, but also register if a vehicle goes on the wrong lane or crosses a red light. If there are consequences to rash driving on roads (like road fines or license suspension), people would be more careful in following street rules. This in turn would reduce the number of accidents on road.

3. System Design

3.1 Design Goals

Following are the design goals for the project:

- Usability – The system should be easy to use for different user groups.
- Compatibility - Throughout the development of the project it must be made sure that the minimum hardware requirements and software versions are compatible.
- Performance – The performance aspect of the system is of utmost importance as the system would be deployed as a real-time solution and the response time and accuracy for detection and tracking should be ideal.
- Efficiency – Reducing the number of resources consumed through design.
- Cost – Reducing the total cost of the project through efficient design.

3.2 System Architecture

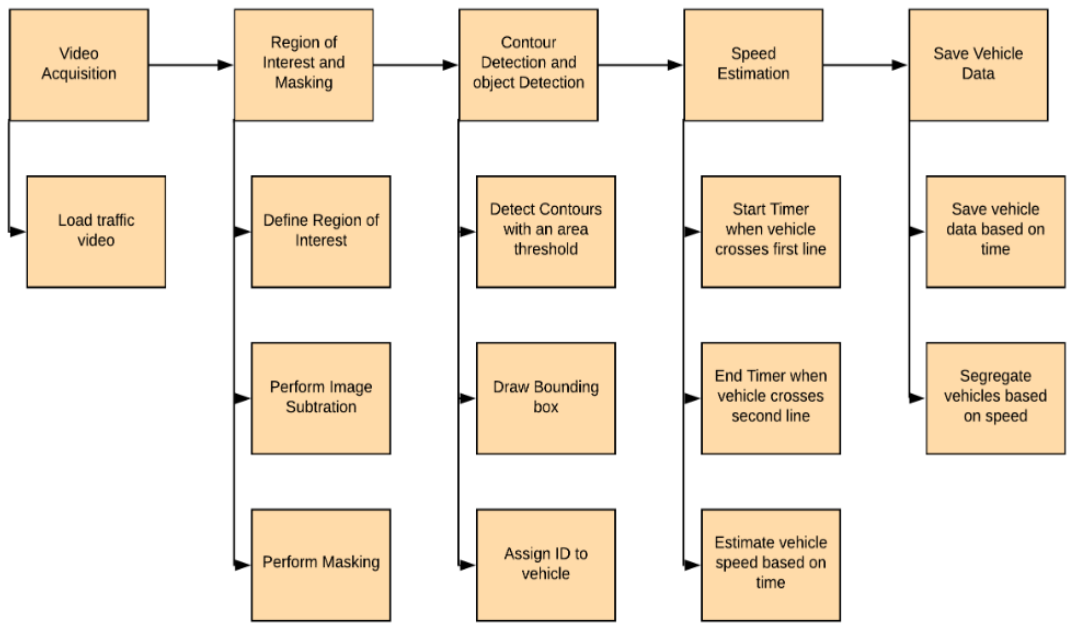

Video Acquisition
Load traffic video
Region of Interest and Masking
Define Region of Interest
Perform Image Subtration
Perform Masking
Contour Detection and object Detection
Detect Contours with an area threshold
Draw Bounding box
Assign ID to vehicle
Speed Estimation
Start Timer when vehicle crosses first line
End Timer when vehicle crosses second line
Estimate vehicle speed based on time
Save Vehicle Data
Save vehicle data based on time
Segregate vehicles based on speed

Fig 2. Project Model

The system architecture is defined in the diagram above, first up we've Video Acquisition and Loading of the video. Next-up we identify our Region of Interest, once the region of interest is identified we perform image subtraction and masking to find the exact location of the object. Once it is done, we do the Object Detection where we draw a bounding box around the object and assign it a Vehicle ID. Next step in the process is to estimate the speed of the vehicle for which we define two reference lines, and we start the timer once the vehicle crosses the first line and stop the timer as soon as it crosses the second line. The speed of the vehicle is calculated using the metrics 'Distance travelled' divided by the 'Elapsed Time'. Once the speed is calculated t\an image of the vehicle is also captured to feed it to a different module that can extract the license plate number of the vehicles captured using the previous module.

3.3 Detailed Design Methodologies

- 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.

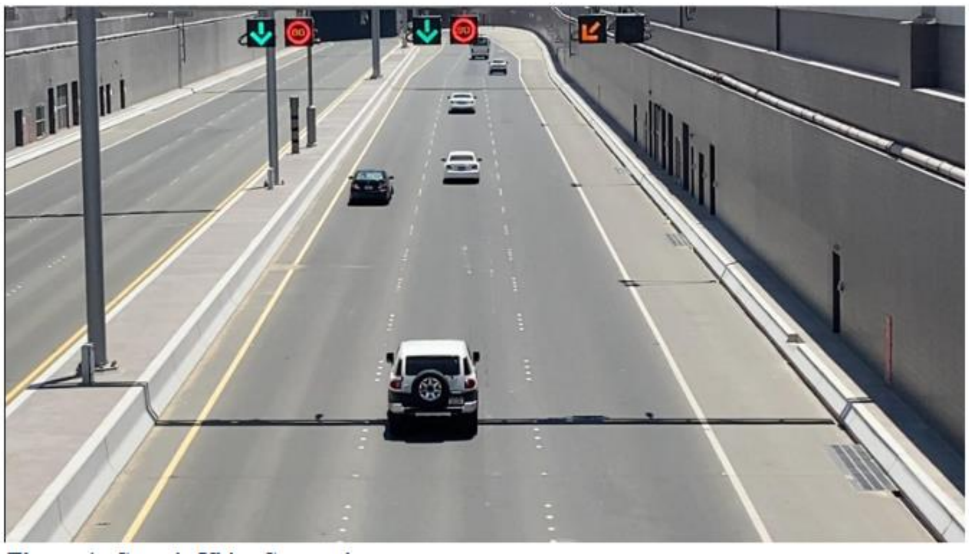


Fig 3. Traffic Feed

- Region of Interest and Masking - 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.

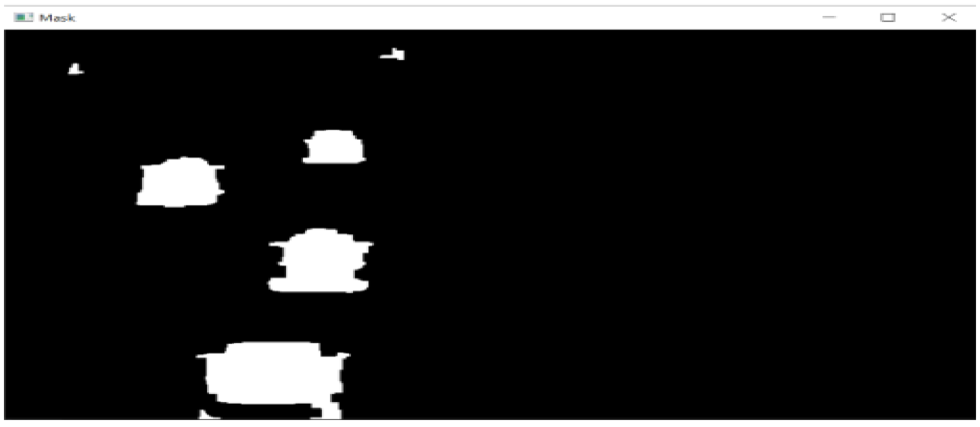


Fig 4. Masking the frames

- Contour Detection and Object Tracking - 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.

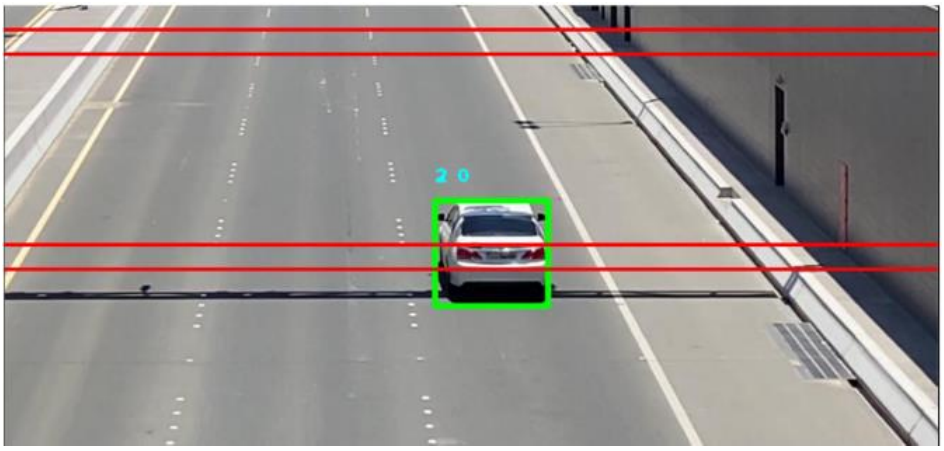


Fig 5. Detection of Vehicle

- 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.

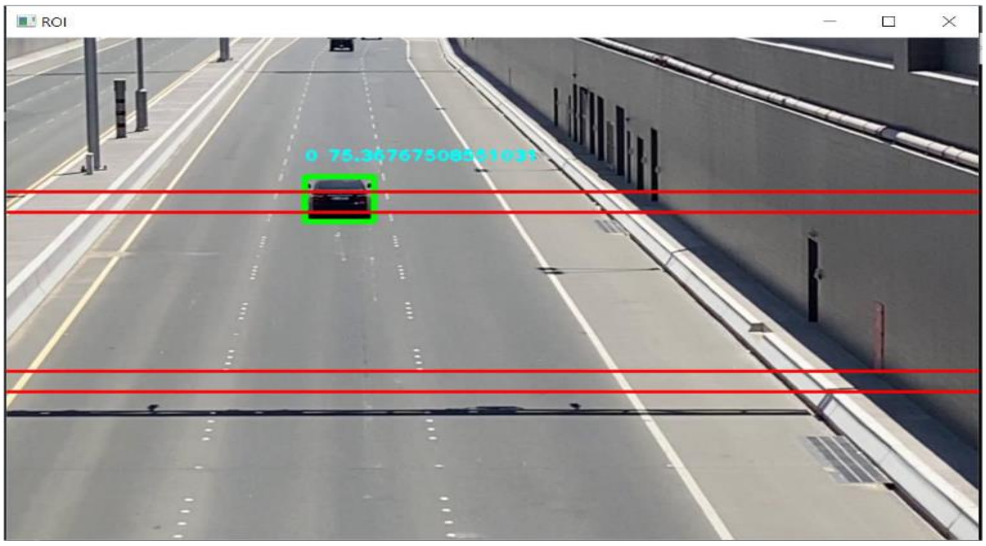
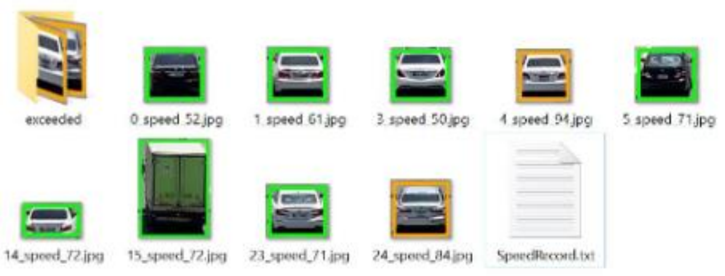


Fig 6. Speed Estimation

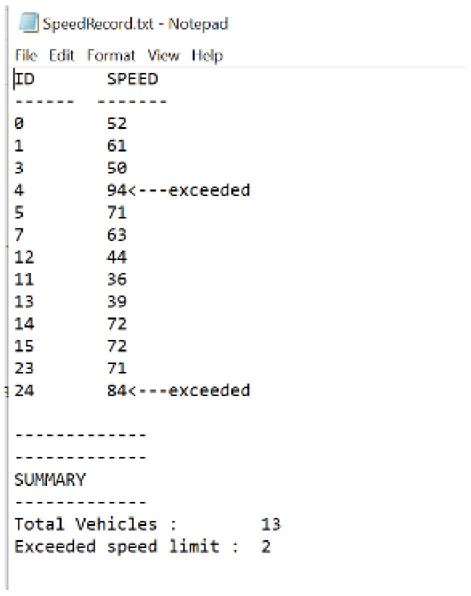
- Save Vehicle Data - The picture of the bounding box (the vehicle) is saved into a file along with the speed. Vehicles crossing the speed limit is segregated



into a separate folder.

Fig 7. Captured Images

- 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



```
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 8. Saved Summary

7. Using Tesseract for License Plate recognition – Further the images saved are to be processed using tesseract to obtain the license plate number of the vehicles.

4. Work done

4.1 Development Environment

4.1.1 Software Requirements:

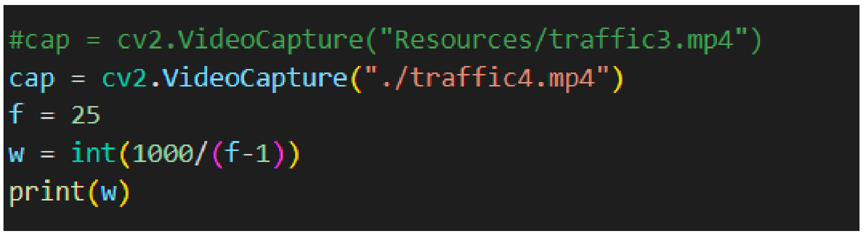
- Python version 3.7
- OpenCV
- Jupyter Notebook
- Google Colab

4.1.2 Hardware Requirements:

- Intel i5 10th Gen or above OR AMD Ryzen 3750H or above
- 16GB RAM or above
- Nvidia GTX 1660ti or above
- Camera recording at 30fps or above

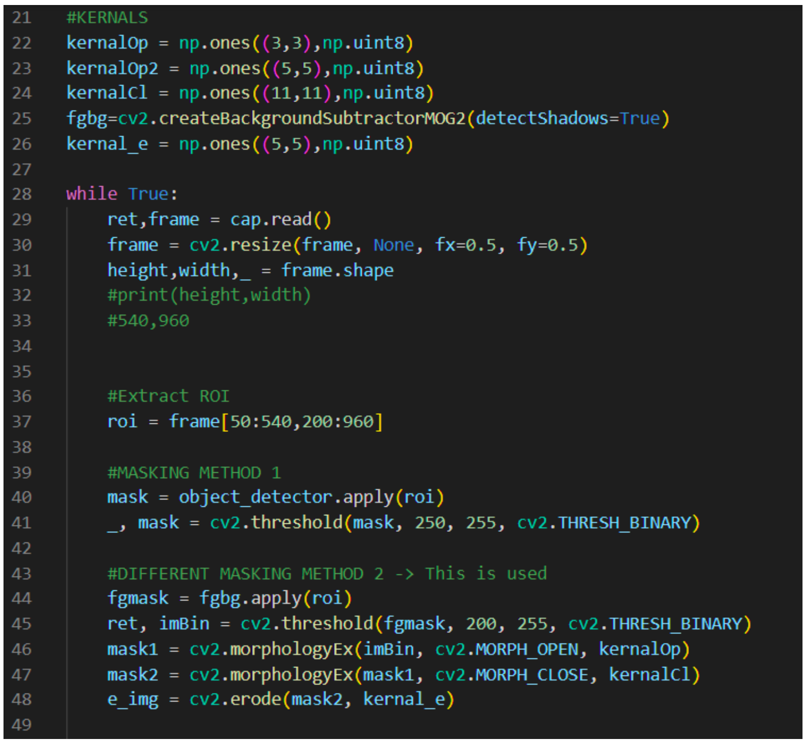
4.2 Implementation

- Video Acquisition



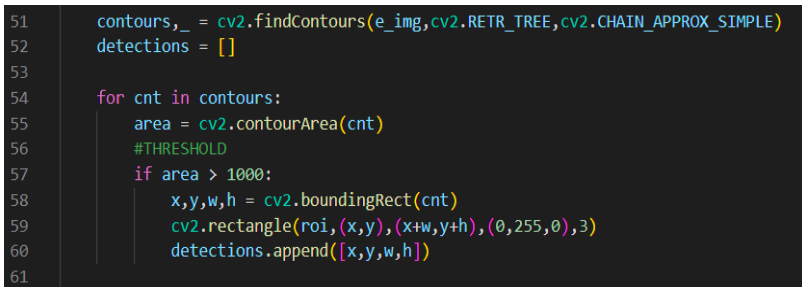
```python
#cap = cv2.VideoCapture("Resources/traffic3.mp4")
cap = cv2.VideoCapture("./traffic4.mp4")
f = 25
w = int(1000/(f-1))
print(w)
```

- Region of Interest and Masking



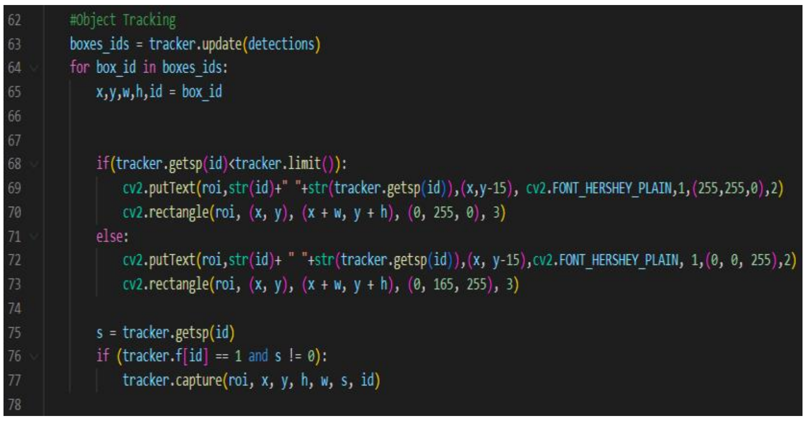
```python
#KERNALS
kernalOp = np.ones((3,3),np.uint8)
kernalOp2 = np.ones((5,5),np.uint8)
kernalCl = np.ones((11,11),np.uint8)
fgbg=cv2.createBackgroundSubtractorMOG2(detectShadows=True)
kernal_e = np.ones((5,5),np.uint8)

while True:
    ret,frame = cap.read()
    frame = cv2.resize(frame, None, fx=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)

```

- Contour Detection

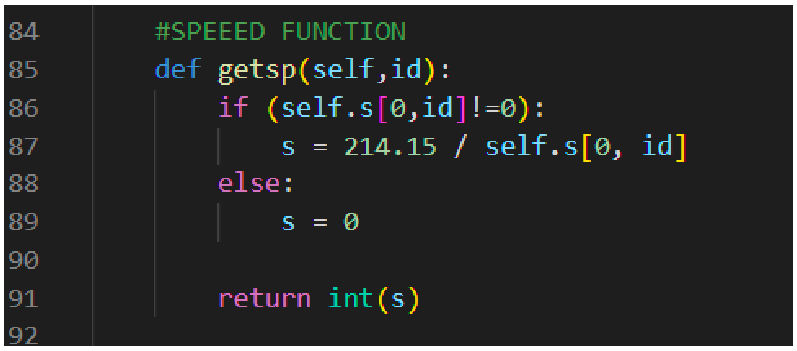


```python
    contours,_ = cv2.findContours(e_img,cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE)
    detections = []

    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



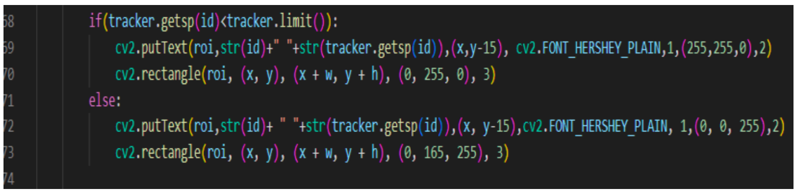
```python
    #Object Tracking
    boxes_ids = tracker.update(detections)
    for box_id in boxes_ids:
        x,y,w,h,id = box_id


        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)

```



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

        return int(s)

```

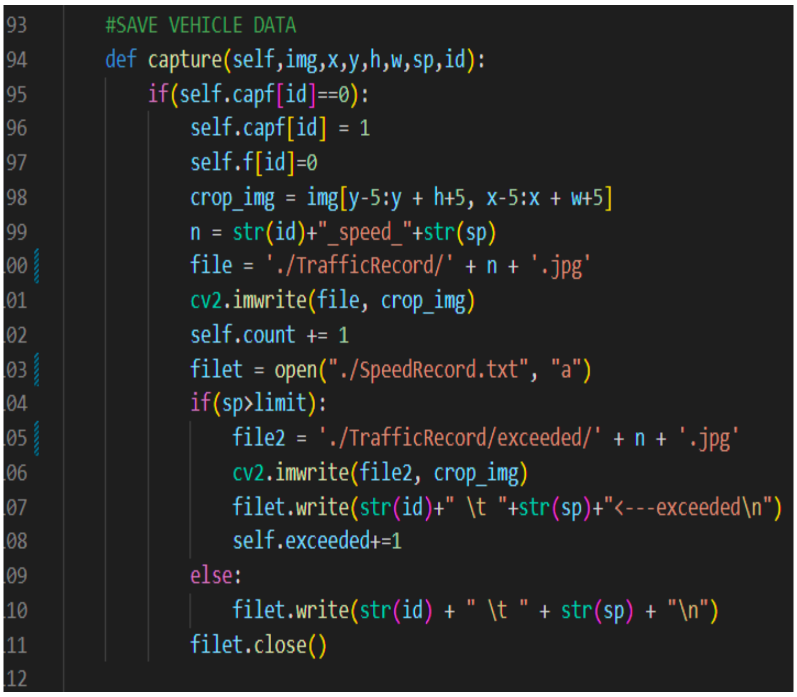
- Speed Estimation

- Drawing Rectangles and displaying on the screen



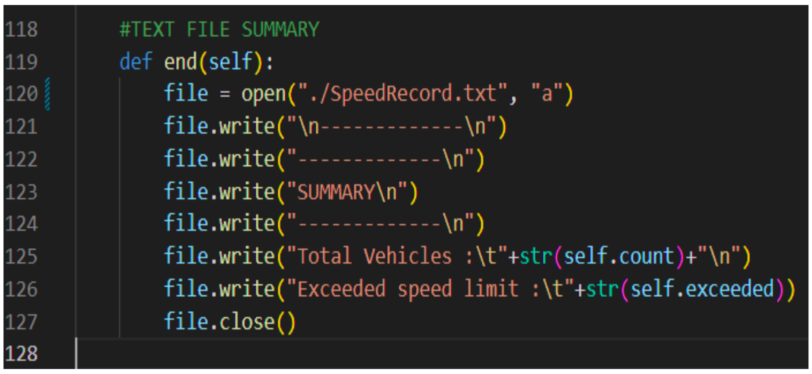
```python
            if(tracker.getsp(id)<tracker.limit()):
                cv2.putText(roi,str(id)+" "+str(tracker.getsp(id)),(x,y-15), cv2.FONT_HERSHEY_PLAIN,1,(255,255,0),2)
                cv2.rectangle(roi, (x, y), (x + w, y + h), (0, 255, 0), 3)
            else:
                cv2.putText(roi,str(id)+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

```

- Save Vehicle Images and speeds



```python
    #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 = './TrafficRecord/' + n + '.jpg'
            cv2.imwrite(file, crop_img)
            self.count += 1
            filet = open("./SpeedRecord.txt", "a")
            if(sp>limit):
                file2 = './TrafficRecord/exceeded/' + n + '.jpg'
                cv2.imwrite(file2, crop_img)
                filet.write(str(id)+" \t "+str(sp)+"<---exceeded\n")
                self.exceeded+=1
            else:
                filet.write(str(id) + " \t " + str(sp) + "\n")
            filet.close()

```

- Create Summary



```
#TEXT FILE SUMMARY
def end(self):
    file = open("./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()
```

4.3 Results and Discussions

Currently the development that has been done so far is satisfactory with an object detector and tracker implemented. The system can identify vehicles passing on a road and track them at the same time along with giving each of the vehicles a unique id that helps in uniquely identifying a vehicle in consecutive frames. Multiple vehicles can be detected, and their speeds can be detected. However, if two vehicles are moving extremely close to each other, it may be detected as a single object. This project requires the camera to be as still as possible, as movement is used to distinguish vehicles from the background

Along with this it is also able to estimate the speed of the vehicle given two reference points. A separate module for license plate recognition has also been set in place which can be used to find the license plate number of the cars. Since the

5. Conclusion and Future Plans

The project is modelled around the field of Computer Vision and makes use video feed to detect the speed of a moving vehicle along with that the license plate number of the vehicle is also logged in case if it must be reported for traffic rule violation. All this is being achieved by making use of for object detection using image subtraction technique, for object tracking we are using a frame-based tracking approach and we track the object in every consecutive frame and OpenCV (for handling the video feed). As of now an object detector and a tracker has been implemented which is capable of tracking vehicles in consecutive frames and along with that it also assigns an ID to each detected object so that it can be uniquely identified.

The module built is also capable of detecting speed of the moving vehicles and log their details in a separate text file along with their corresponding IDs and Speed. It also captures the images of the vehicles once it passes a reference line.

A separate module for license plate detection is also built which makes use of the captured images to extract the license plate number of the vehicles and log them in a csv file.

6. References

Websites

[1] Adrian Rosebrock (2019, December, 02). OpenCV Vehicle Detection, Tracking, and Speed Estimation [Online]. Available: https://www.pyimagesearch.com/2019/12/02/opencv-vehicle-detection-tracking-and-speed-estimation/

[2] Pysource (2021, October, 2021). Detect vehicles speed from CCTV Cameras with Opencv and Deep Learning[Online]. Available: https://www.youtube.com/watch?v=j10j8IuKSBI

Research Papers

[3] Hector Mejia, Esteban J. Palomo, Ezequiel Lopez-Rubio. ‘Vehicle Speed Estimation Using Computer Vision And Evolutionary Camera Calibration’.

[4] Awad Abdelhalim, Montasir Abba, Bhavi Bharat Kotha, Alfred Wick, ‘A Framework for Real-time Traffic Trajectory Tracking, Speed Estimation, and Driver Behavior Calibration at Urban Intersections Using Virtual Traffic Lane’