Video Surveillance System: Vehicle Identification and Speed Detection

[1]Prof. Saurabh Patil, [2]Hemant Gosavi, [3]Mukesh Jain, [4]Jenish Paul

[1]Professor, [2]Student, [3]Student, [4]Student,
[1]Department of Computer Engineering,
[1]Xavier Institute of Engineering, Mumbai, India.

Abstract : The value of data has increased exponentially in recent years. Organizations are using online data to enhance their product and data interpretation. If traffic data from cars on the road can be easily obtained from videos, it can be very useful. In contrast to regular data traffic, surveillance videos contain an enormous volume of data that remains unprocessed, resulting in information loss. The aim of our project is to develop a method for retrieving vehicle data from videos that is both efficient and reliable. The vehicle's license plate number, colour, manufacturer's name, and time-stamp are all included in this information. The information will then be used for a variety of purposes. This information would aid in a more in-depth examination of road traffic. This research would aid in the gathering of information that will aid in the improvement of road transportation in the future. Additionally, since the data is in text format, searching and altering the data can be both quicker and quicker. This altered data may be correlated with video data if necessary. Without the text data, the above procedure will need less scrubbing through the video to achieve the desired stage. Furthermore, data stored in text format is much smaller than data stored in video format. Our project's implementation would contribute to the development of a smart traffic grid. We get the details we need one at a time. To begin, the vehicle is identified using YOLO (You Only Look Once). The requisite data is then retrieved from each vehicle's photographs. The vehicle's license plate is first detected. The license plate after some pre-processing.

IndexTerms **– Speed Detection, YOLO, Surveillance.**

I. INTRODUCTION

With the concept of advancement of growth in technology and transmuting of smart cities there's a need to make traffic system more indispensable component among the society. vehicles and infrastructure limitations, this is a difficult challenge that necessitates the use of specialised algorithms as well as detailed traffic details. The number of cars, their licence plates, the colour of the car, and the type of the car are all data that can be useful for traffic monitoring and law enforcement work. We will use traffic data to not only improve the reliability of traffic control, but also to adjust management policies to changing circumstances and catch people who are breaking traffic laws. Traffic violation monitoring, collision identification, car counters, and classifiers are all part of the integrated traffic control scheme. Both of these feed real-time data into a Vehicle Control Room, enabling traffic authorities and users to get alerts in real time.

1.1 AIMS AND OBJECTIVES

The primary goal of this paper is to provide solutions to capture important vehicle features from vehicle footages. The paper will present the concept, technology stack, components and the outcomes of implementing the solution. Apart from vehicle details extraction we also try to implement speed detection. By improving the digitization in the traffic system, we aim to help the law enforcement to make the roads a much safer place

II. REVIEW OF LITERATURE

2.1 Approach 1

The government have placed traffic police so that they can keep a track on the speed limit of a vehicle ,whether a person follows a traffic rules or not. But this method is time consuming and hardware costly and also requires a lot of man power which is not so efficient at last. Currently to capture and detect the speed of the moving vehicle speed camera are used in various parts of country including India. It is used for recording fast-moving objects like photographic images onto a storage medium.

Figure 2.1

2.2 Approach 2

2.2.1 Vehicle Data Extraction

To eliminate the reliance on an external source for data extraction, we can extract the required data directly from the vehicle image itself. For this, we have divided vehicle data extraction into four parts. They include Vehicle detection, License Plate detection, Logo detection, and Colour detection. Each of these detections runs one after the other and the extracted information is stored in a CSV file.

2.2.2 Vehicle Speed Detection

Video Surveillance is a very popular research topic in computer vision applications that continuously tries to detect and track down the targets. In our proposed system ,the speed of the vehicle is detected with the help of initially converting a video to frames and after that applying some kind of vehicle detection algorithm on each frame and assigning them with unique id so it's easier to track them down. And for speed calculation 15we have drawn 2 lines at the centre of road. As soon as car crosses the 1st line, we note down the co-ordinates of the car and in upcoming frame we again note down co-ordinates of vehicle having same ID so by applying speed=distance between pixels in meter multiplying that by FPS and 3.6(to covert to km/h) we are able to generate the speed of each car and show down

III. DESCRIPTION

In this digital era, data is a valuable commodity. Details can be lost if some type of data is left unprocessed. The aim of this paper is to present a method for extracting data from vehicle video. This information can be analysed to improve law enforcement and make the highways a safer location. Searching and filtering by words would be possible because the data has been transformed from video and text. In addition, the amount of storage space available is significantly reduced. The main aim of this paper is to present strategies for extracting essential vehicle characteristics from vehicle video. The design, technology stack, elements, and results of implementing the solution will all be presented in this document. We are also attempting to introduce speed detection in addition to extracting vehicle data. The implementation of an "Intelligent Traffic System" would be aided by the combination of each of these modules.

3.1 Analysis

3.1.1 Data Flow Diagram

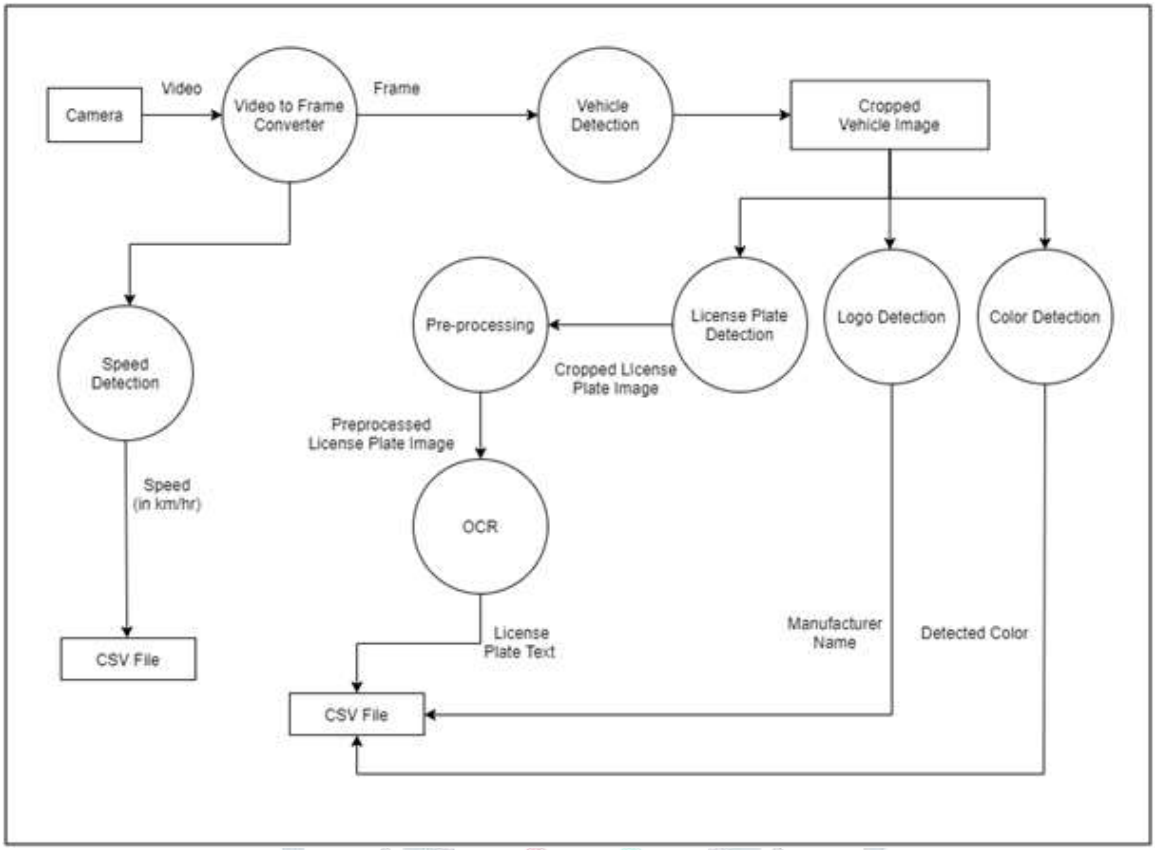

Figure 3.1 DFD of our Proposed System

Data Flow Diagram represents the flow of data through a process or a system. A Data Flow Diagram provides information about the outputs and inputs of each entity and the process itself.

3.2 Design

3.2.1 Flowchart of whole system

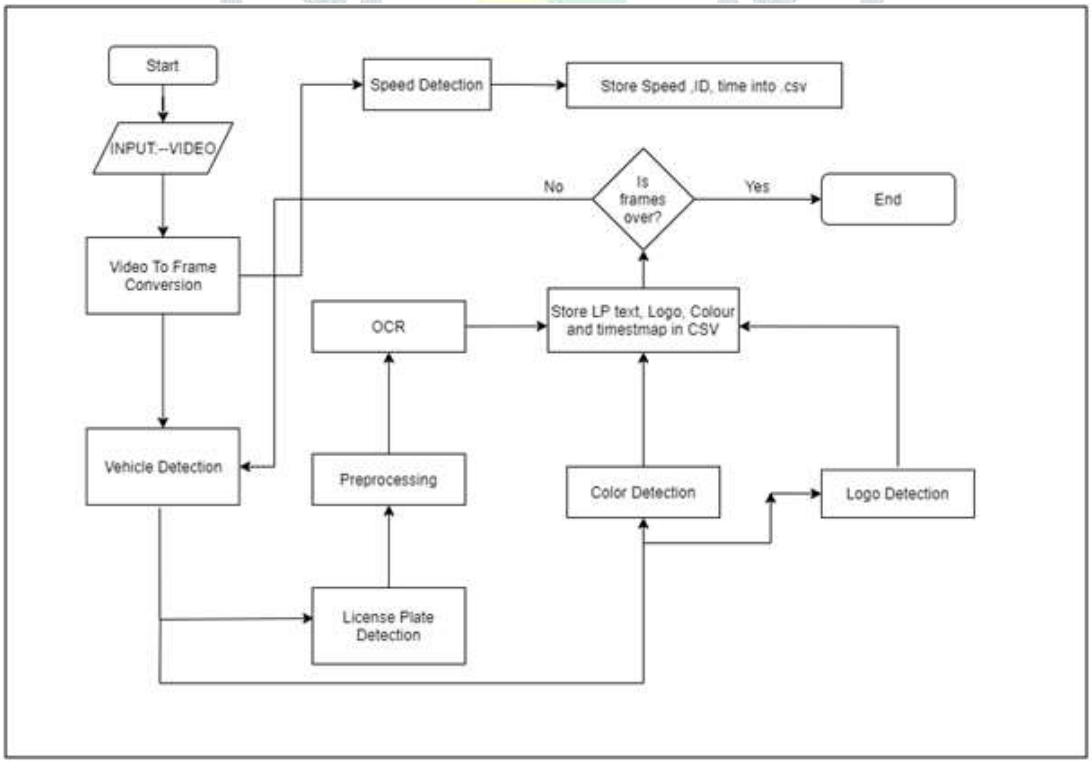

Figure 3.2: Flowchart for our project

The above flowchart depicts the flow of the system. The system starts by taking the video as input and converts it into frames. Then each frame is sent for vehicle detection, if there are vehicles then the cropped image sent for license plate detection. Every

license plate image is pre-processed to increase OCR accuracy. Then the license plate text is extracted using OCR. Then the colour detection and logo detection are performed. and all these results are stored in the CSV file. For speed detection, each frame is given for detection, tracking and finding of speed of the vehicle and displaying to the user. At last, all 3 parameters id of car ,speed of vehicle and timestamp with reference to video is stored into a single user and at last user can download that csv and use it for further analysis.

3.2.2 Flowchart detailing the speed detection process

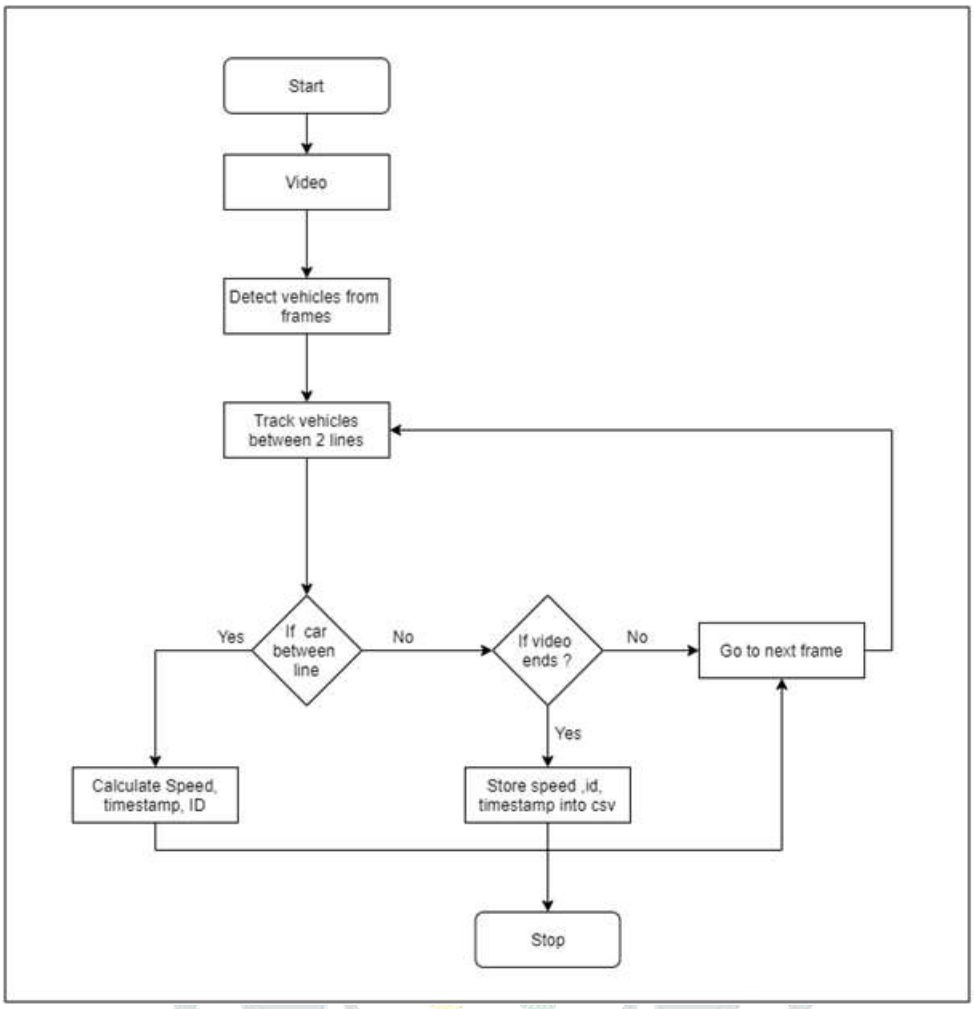

Figure 3.3: Flowchart explaining the speed detection process

For speed detection, we take video and convert it into number of frames and on each vehicle, we apply harr cascade classifier for vehicle detection and if we find any vehicles then we try to track those vehicles with the help dlib tracker. And calculate timestamp, id, speed of vehicle and store it into csv. If not find then move to next frame. And if frames end, then stop vide.

3.2.3 Sequence Diagram

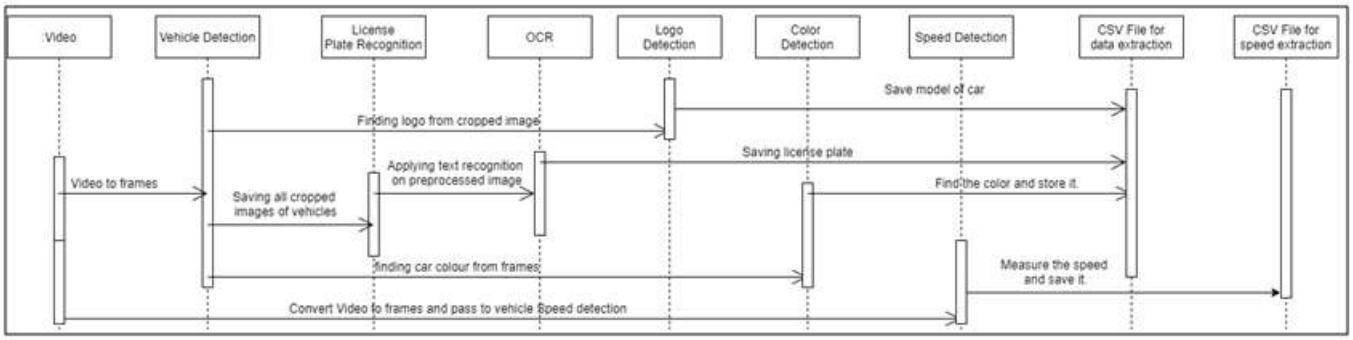

Figure 3.4 Sequence

The video is the system's initial object. The entire procedure begins with the recording. Vehicle identification, licence plate recognition, OCR, emblem detection, colour detection, speed detection, and a CSV file are the other objects in the sequence diagram. The process begins with video, which is translated into frames and fed into a vehicle detection object, which detects vehicles in each frame and stores them as images, before being fed into a speed detection object. Following that, the performance of vehicle detection is given to licence plate detection from an image, which is then given to OCR. The picture of the detected vehicle is also subjected to logo detection in order to obtain the vehicle's logo. Similarly, it is given to a colour detection object in

order to determine the colour of a vehicle. Now, OCR is used to locate the exact number plate written on the licence plate using the output from licence plate recognition. All of the OCR, Logo Detection, Color Detection, and Speed Detection outputs are stored as a CSV file, which can be used for analysis and further work if necessary. The video is converted to frames for the speed detection part, and each frame is given to the vehicle speed detection algorithm, which is responsible for calculating the speed of each vehicle and storing the id, timestamp, and speed into a CSV file that can be used later.

3.3 IMPLEMENTATION METHODOLOGY

3.3.1 Vehicle Data Extraction

3.3.1.1 Introduction to YOLO

As mentioned before Vehicle detection, License Plate detection and Logo detection are implemented using YOLO, precisely version 4 or YOLOv4. Out of various implementation of YOLOv4 we have used python module 'yolov4' implementation. We had previously used darknet and PyTorch implementation of YOLO but we finalized on using 'yolov4' for the following reasons: Linearity across all three detection modules. The working can be fine-tuned to our needs by changing the necessary files. Detections from these modules can be integrated with Flask.

After a detection is complete the output of the detection is obtained in the format shown below. [x, y, w, h, class id, confidence]

x : X co-ordinate of top-left pixel

y : Y co-ordinate of top-left pixel

w : Width of the detection

h : Height of the detection

class id : Class ID of the object detected

confidence : Confidence of detection

3.3.1.2 Vehicle Detection

Initially we used darknet implementation of YOLO for vehicle detection. Since the pretrained weights provided by YOLO weights were sufficient for vehicle detection it was not necessary to train a custom detector. Output of darknet YOLO is given below.

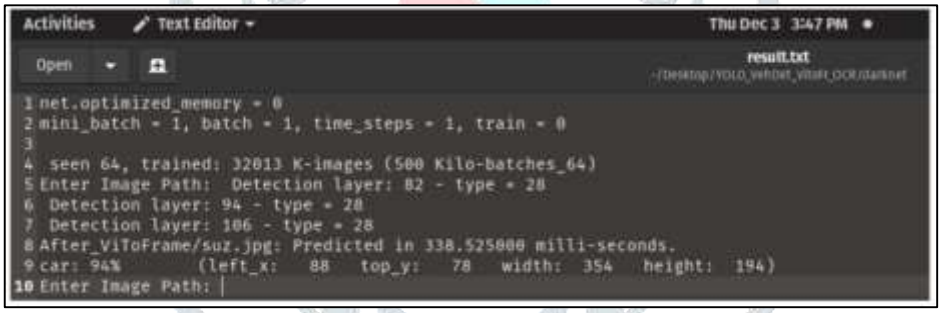

Figure 3.1: Output text file given by darknet implementation of YOLO

Figure 3.2: Bounding Box drawn using the co-ordinates given by YOLO detection

As one can see the darknet implementation gives a text file as an output. In order to crop the vehicle image, we had to write a secondary program to read through the output file to obtain the co-ordinates and crop the image. As one can see this is unnecessary overhead. This unnecessary step is eliminated by using 'yolov4' module. Using 'yolov4' module we get the detection output in the form of an array as mentioned before. It contains the necessary co-ordinates for cropping and also the class id for identifying the object and it also holds the confidence of detection. As a result, while using 'yolov4' module reading through external file is not required. Using the co-ordinates obtained .Before we crop the ROI and obtain the vehicle image which is passed for License Plate Detection, Logo Detection and Colour Detection.

3.3.1.3 License Plate Detection

The pretrained YOLO weights are not capable of detecting License Plate. As a result, a custom, YOLO detector had to be trained. In order to start training we need the dataset containing images with license plate and for each of the image there needs to be a txt file with co-ordinates (annotations) of the license plate. One of the good places to obtain dataset for YOLO training is 'Open Images Dataset' (https://storage.googleapis.com/openimages/web/index.html). Here we search for the classes we require (i.e License Plate) and download the images and annotations files using OIDv4 ToolKit.

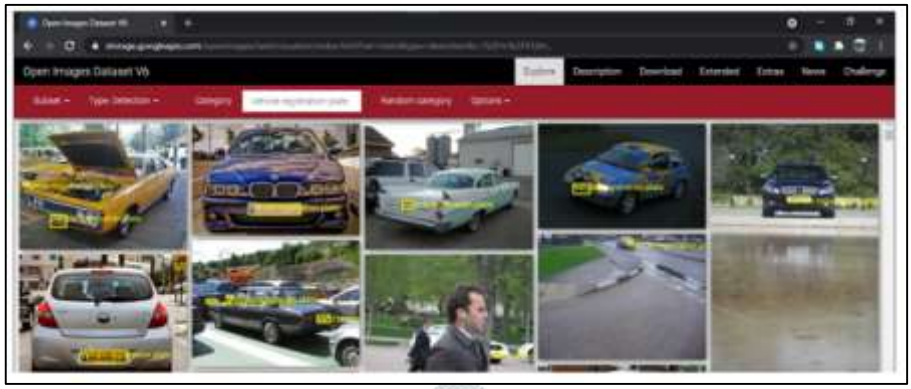

Figure 3.3: Vehicle registration plate Dataset Images

'Vehicle registration plate' is name of the category we needed so we downloaded it using 'OIDv4 Toolkit'

Figure 3.4: Downloading Dataset

Firstly, the license plate is binarized. In binarization we have kept a pixel white if its below certain threshold and the pixel is black if it is above the threshold. After binarization we obtain a clean black and white image. Even after binarization the image contains jagged edges. This is removed by rescaling the image by a factor of 5. Lastly the image is smoothened using Gaussian Blur.

Figure 3.5: Pre-processing on License Plate

After pre-processing, OCR is applied on the image using pytesseract and the OCR output is validated to check if the OCR output matches with some pre-defined License Plate format. The OCR output is rejected if: The starting two characters do not match any of the LP Abbreviations of Indian states.(Eg. MH,AP,etc.) It contains any lowercase letters. It does not end with a number. If the OCR output passes the above validation checks it is stored in CSV file.

3.3.1.4 Logo Detection

The pretrained YOLO weights are not capable of detecting Logo. As a result, a custom YOLO detector had to be trained. One of the problems faced for this part was that even dataset was not available for Logo detection to perform YOLO training. As a result, we had to create dataset consisting of major car manufactures in India. The dataset had to comply with the format required by YOLO i.e each imaged should be annotated. The annotation co-ordinates and the class ID are stored in a text file with same name as the image. After the dataset is created, we trained the custom detector using Google Colab with GPU Runtime for

faster training. The Logo detector gets the cropped car image as input and gives the bounding box co-ordinates, confidence and class ID as output. The class ID is used to get the manufactures name which is then stored in the CSV file.

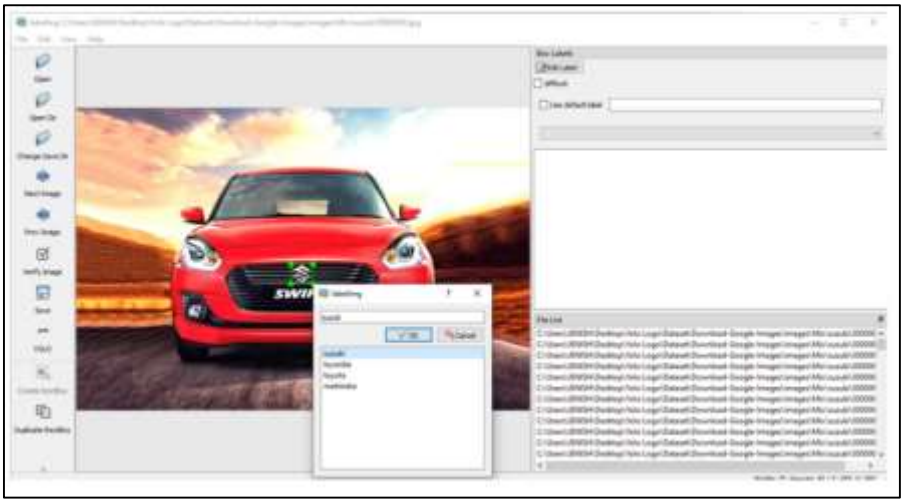

Figure 3.6. Annotating Image for YOLO

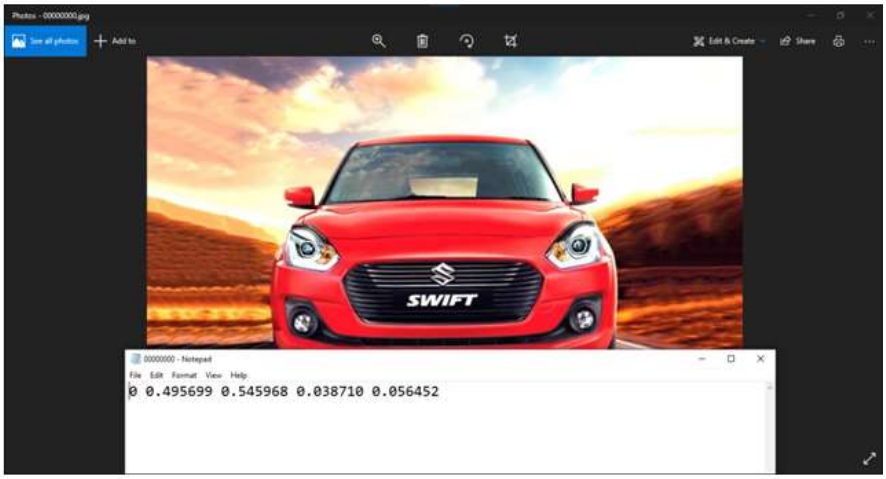

Figure 3.6: Annotated Image with corresponding text file

3.3.1.5 Colour Detection

3.3.1.5.1 Approach 1: Pixel Method to Get Prominent Color.

We have used the PIL library to do it. Firstly Creating 9 class for colour then getting RGB value of every pixel present in image. Find count and strength of colour in image. We have taken count of all pixel and store them in dictionary and with value¿10 we find sum of all that pixel into the image. Now that pixel is given to color heel function which is responsible for estimating the colour of that pixel. So no every pixel is associated with 3 classes and one value.

1.Estimate Color of pixel 2.Result: It will show all colour present with their percentage.

```
    if Colors.Red in color_classes and Colors.Green in color_classes:
        return Colors.Yellow(dominant_colors[0].value)
    elif Colors.Red in color_classes and Colors.Blue in color_classes:
        return Colors.Pink(dominant_colors[0].value)
    elif Colors.Blue in color_classes and Colors.Green in color_classes:
        return Colors.Teal(dominant_colors[0].value)
elif total_colors == 3:
    if dominant_colors[0].value > 200:
        return Colors.White(dominant_colors[0].value)
    elif dominant_colors[0].value > 100:
        return Colors.Gray(dominant_colors[0].value)
    else:
        return Colors.Black(dominant_colors[0].value)
else:
    print("Dominant Colors : %s" % dominant_colors)
```

Figure 3.8: Loading screen while data extraction is in progress.

Figure 3.9 Colour Detection Output

3.3.1.5.2 Approach 2:Using Pixel Method + Dark Channel Dehazing

A large part of this project will discuss the removal of haze from pictures. You can apply this algorithm to several other problems as well. To provide us with clearer images, this algorithm uses ambient light. We are encountering more air and light pollution as cities grow larger. Both lead to hazy images that, in real time, are difficult to process. To produce a clear picture, we will be using transmission maps and atmospheric light.

Before we use Dark Tube, which is an assumption based on their haze-free outdoor image analysis. It represents the finding that at least one color channel(R/G/B) has pixels with very low intensity in most non-sky pictures. Now, to measure the ambient light, we first identify the dark channel's top 0.1 percent brightest pixels. And, the highest value is then known as the light of the atmosphere. A coarse map of a local area is then defined, after which we create an image edge information map that is then combined to create an accurate transmission map for each pixel representing fog density. To resolve this problem, we use a simplified process. To represent the foggy image's edge detail details, a fine map with minimum values for the R, G and B channels is derived. Now we created four functions that give us clear images as outputs:

- atmospheric light: Finds the atmospheric light by scanning through the image.
- Dark channel finds: Finds the dark channel for each pixel in the image.
- Coarse: Finds the coarse map for image
- Dehaze: Uses input from the 3 functions mentioned above and generates a dehazed image which is finally used to train the models.

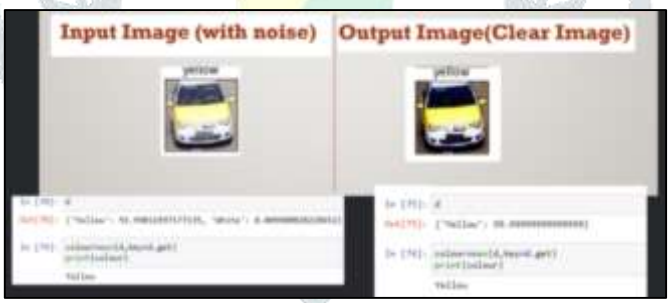

Figure 4.10: Comparison of Colour Output

3.3.2 Vehicle Speed Estimation

It involves three steps to find out speed of vehicles using cameras:

- Vehicle Detection
- Vehicle tracking
- Speed Calculation

Initially we have to convert video to frames and identify vehicles from frames and then between 2 lines marked on the road we can track down the vehicle and note down their positions and calculate speed of each vehicle with the help of speed formula and lastly store all the basic info into a csv file so it's easier to track vehicles and their speed on the road passing-by.

3.3.2.1 Vehicle Detection

Haar Cascade Classifiers : We will implement our use case using the Haar Cascade classifier. Haar Cascade classifier is an effective object detection approach which was proposed by Paul Viola and Michael Jones in their paper, "Rapid Object Detection using a Boosted Cascade of Simple Features" in 2001. So, let's try to understand what these Haar Cascade Classifiers are. This is basically a machine learning based approach where a cascade function is trained from a lot of images both positive and negative. Based on the training it is then used to detect the objects in the other images. So how this works is they are huge

individual .xml files with a lot of feature sets and each xml corresponds to a very specific type of use case. For our project we have used pre-trained Haar Cascade Classifier since it was available online and it was working great with project so we didn't go on training a new one.

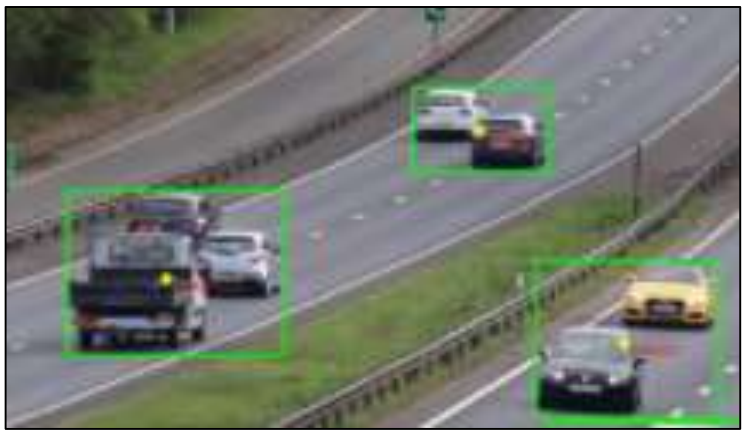

Figure 3.12 Harr cascade Classifier

3.3.2.2 Vehicle Tracking

In our project case we have tracked vehicle using co-relation tracker from dlib library which is simpler and good method s compared to centroid tracking. The dlib correlation tracker implementation is based on Danelljan et al.'s 2014 paper, Accurate Scale Estimation for Robust Visual Tracking. Their work, in turn, builds on the popular MOSSE tracker from Bolme et al.'s 2010 work, Visual Object Tracking using Adaptive Correlation Filters. While the MOSSE tracker works well for objects that are translated, it often fails for objects that change in scale. The work of Danelljan et al. proposed utilizing a scale pyramid to accurately estimate the scale of an object after the optimal translation was found. This breakthrough allows us to track objects that change in both (1) translation and (2) scaling throughout a video stream — and furthermore, we can perform this tracking in real-time.

```
tracker = dlib.correlation_tracker()
tracker.start_track(image, dlib.rectangle(x, y, x + w, y + h))

carTracker[currentCarID] = tracker
carLocation1[currentCarID] = [x, y, w, h]

currentCarID = currentCarID + 1
```

Figure 3.13: Code for Tracking Vehicle

For our project with the help of dlib library we have tried to assign a unique id to every car and store it so it's easier to know the vehicles in video with the help of id assigned to each image. Again, with the help of dlib we are able to store co-ordinates of vehicle from video so it will be helpful further for speed calculation.

3.3.2.3 Speed Calculation

We are calculating the distance moved by the tracked vehicle in a second, in terms of pixels. With distance travelled per second in meters, we will get the speed of the vehicle. Two reference lines have been set, one for vehicle entry and one for exit. Initially with the help of dlib tracker we calculate 2 different co-ordinates of car w.r.t to their id and store them in [x1,y1,w1,h1] and [x2,y2,w2,h2]

```
if frameCounter % 1 == 0:
    [x1, y1, w1, h1] = carLocation1[i]
    [x2, y2, w2, h2] = carLocation2[i]
```

Figure 3.14: Starting and Ending Co-ordinates of Vehicles

Figure 3.14: Process of Speed Calculation

As soon as car crosses 2nd line marked on road, we are able to generate speed with the help of d pixels formula which is distance between pixels. d pixel gives the pixel distance travelled by the vehicle in one frame of our video processing. To estimate speed in any standard unit first, we need to convert d pixels to d metres. Now, it is the time to deliver the final blow (speed = d_meters * fps * 3.6). d_meters is the distance travelled in one frame. We have already calculated the average fps during video processing. So, to get the speed in m/s, just (d metres * fps) will do. We have multiplied that estimated speed with 3.6 to convert it into km/hr. After doing all this we are able to get speed of the car and store the id, speed of car ,and timestamp when it crossed according to video into a single csv file.

IV. TESTING AND RESULTS

4.1 Testing

4.1.1 Testing YOLO Detection

Figure 4.1 Frame 5 from Input Video

Figure 4.2 Detections on Frame 5

Figure 4.3 Frame 33 from Input Video

Figure 4.4 : Detections on Frame 33

As we can see YOLO successfully detects License Plate and the Logo(shown in the bounding box). Also, vehicle detection is also performed by YOLO in the backend.

4.1.2 Testing OCR

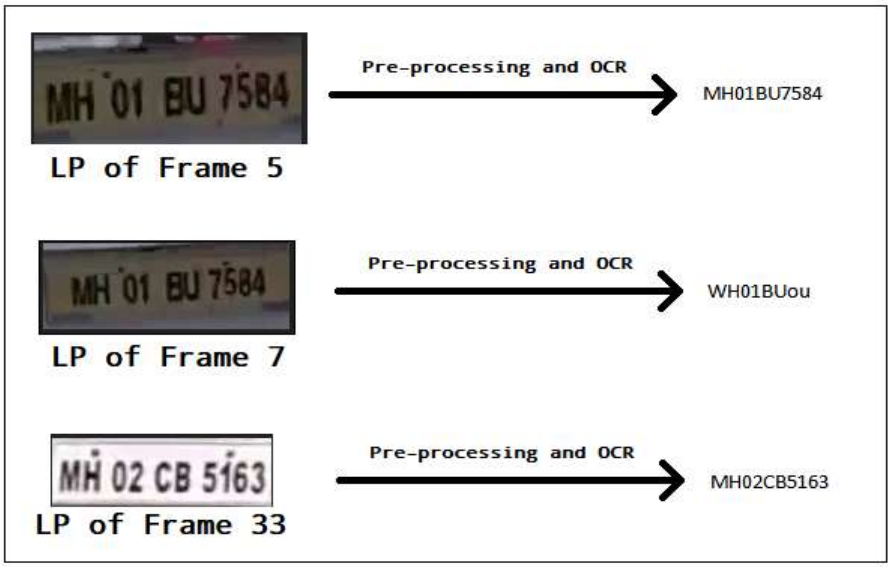

Figure 4.5: OCR Output of LP(License Plate)

The OCR output from Frame 5 and Frame 33 are accurate. The OCR output from Frame 7 is not correct and after passing it through License Plate Format Validation it will be rejected and will not appear in final output.

4.2 RESULTS

4.2.1 Screenshots of the Web Application:

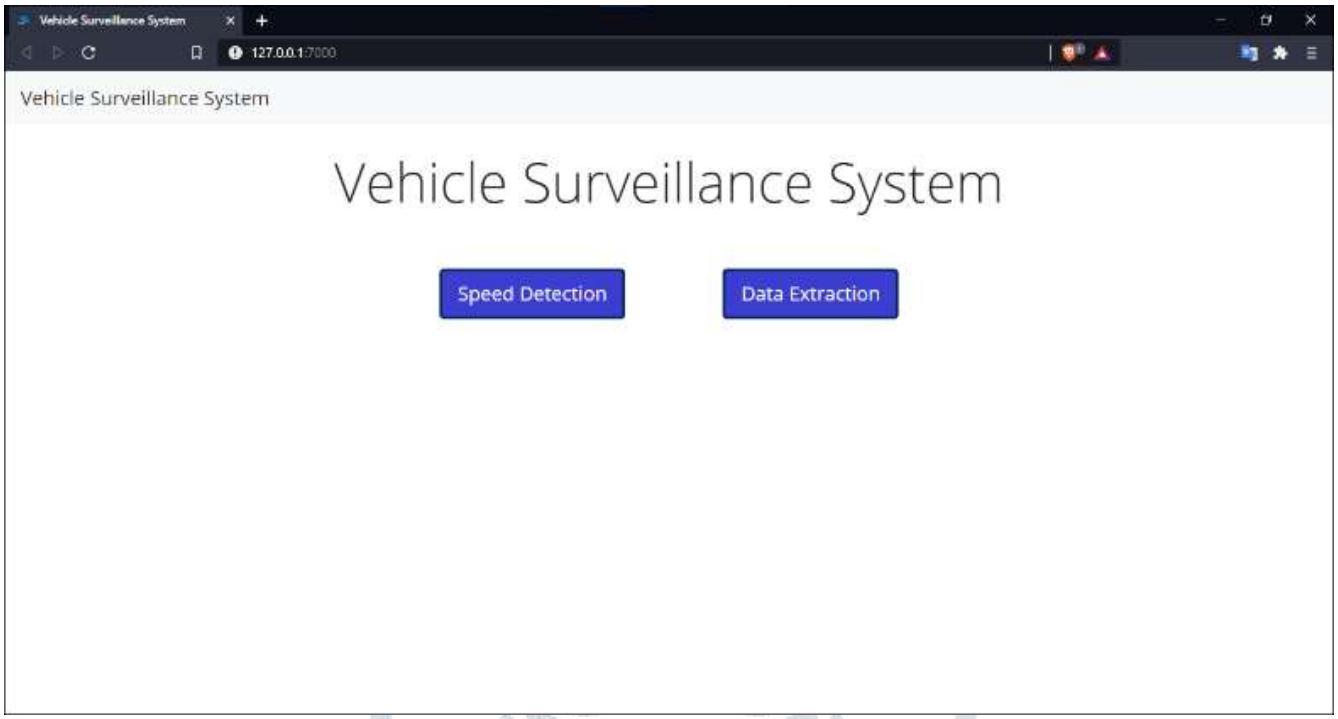

Figure 4.6: Home Page

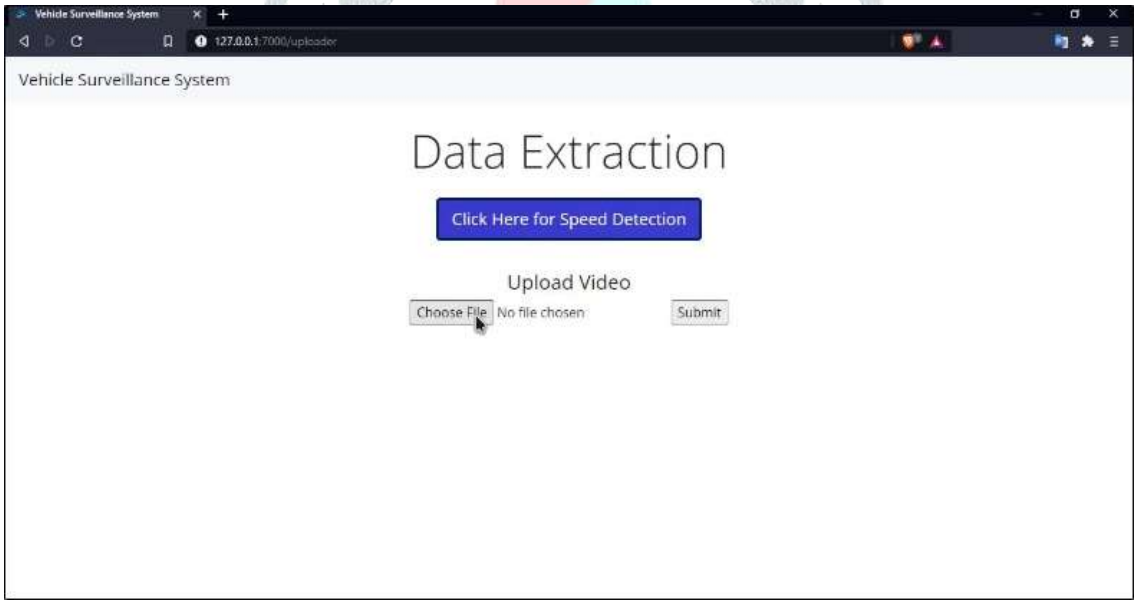

Figure 4.7: Vehicle Data Extraction

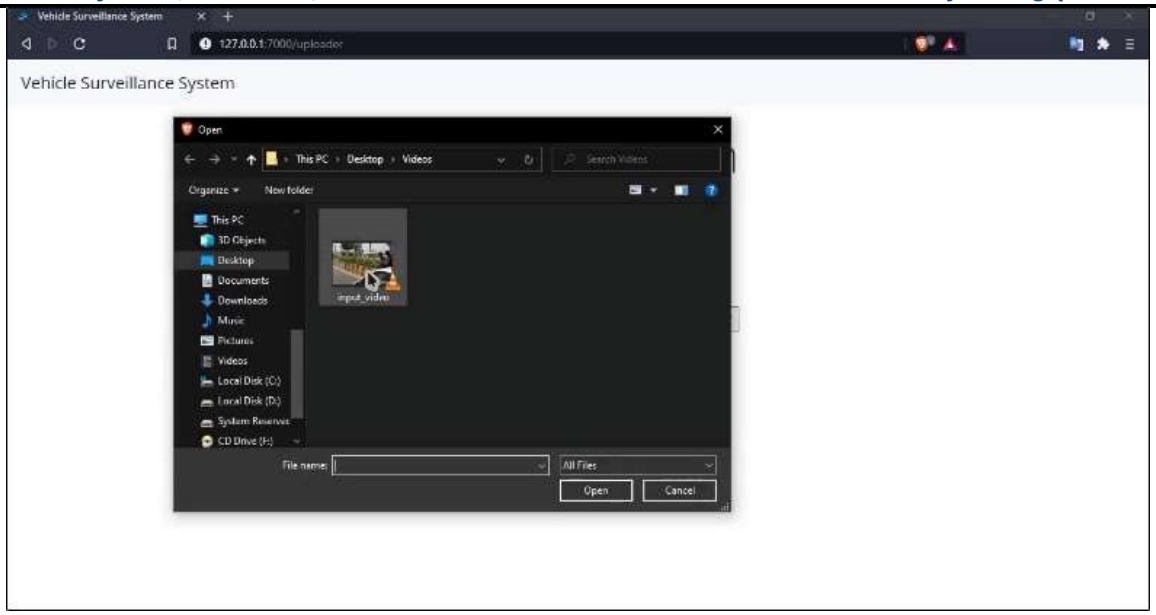

Figure 4.8 : Uploading Video for Data Extraction

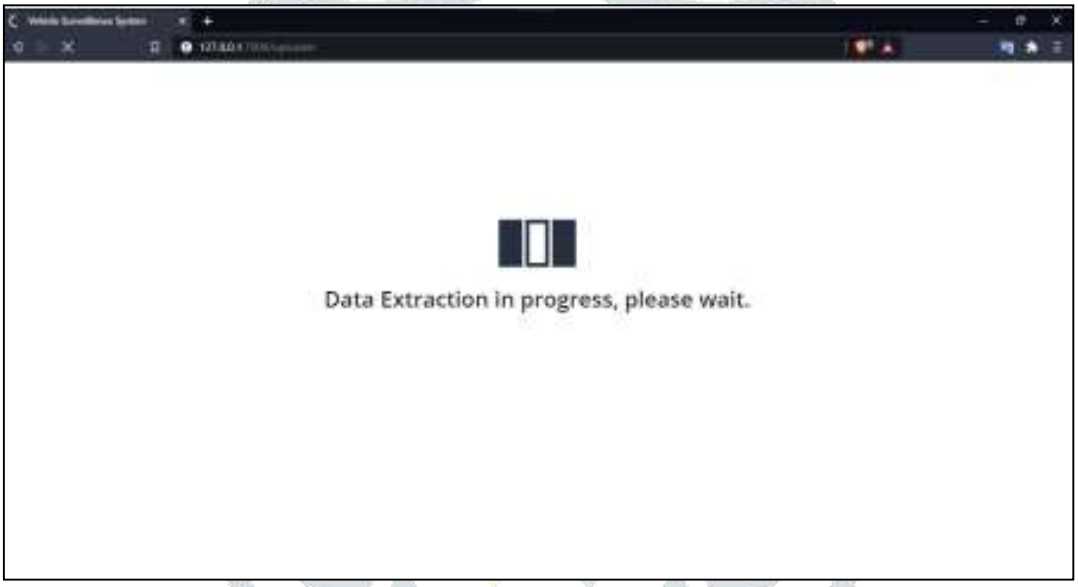

Figure 4.9: Loading screen while data extraction is in progress.

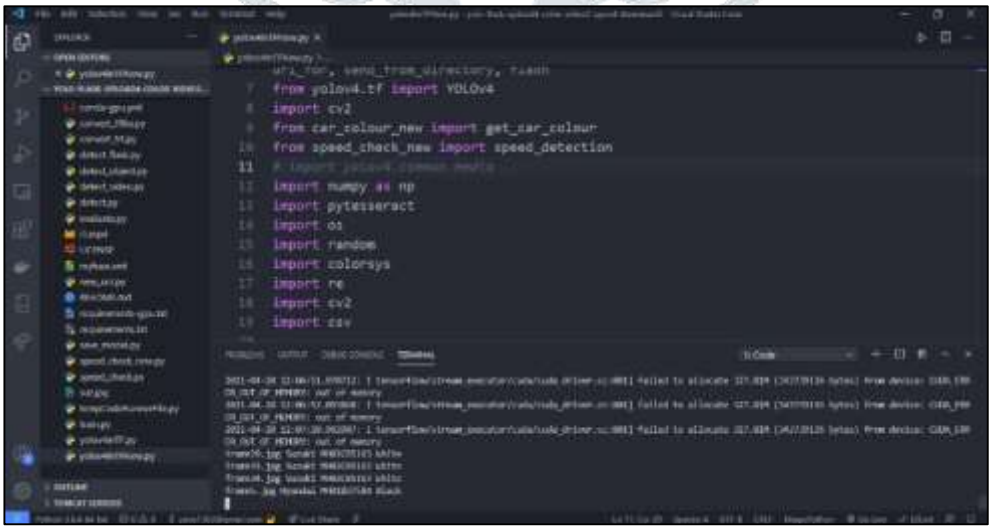

Figure 4.10: Results printed at backend

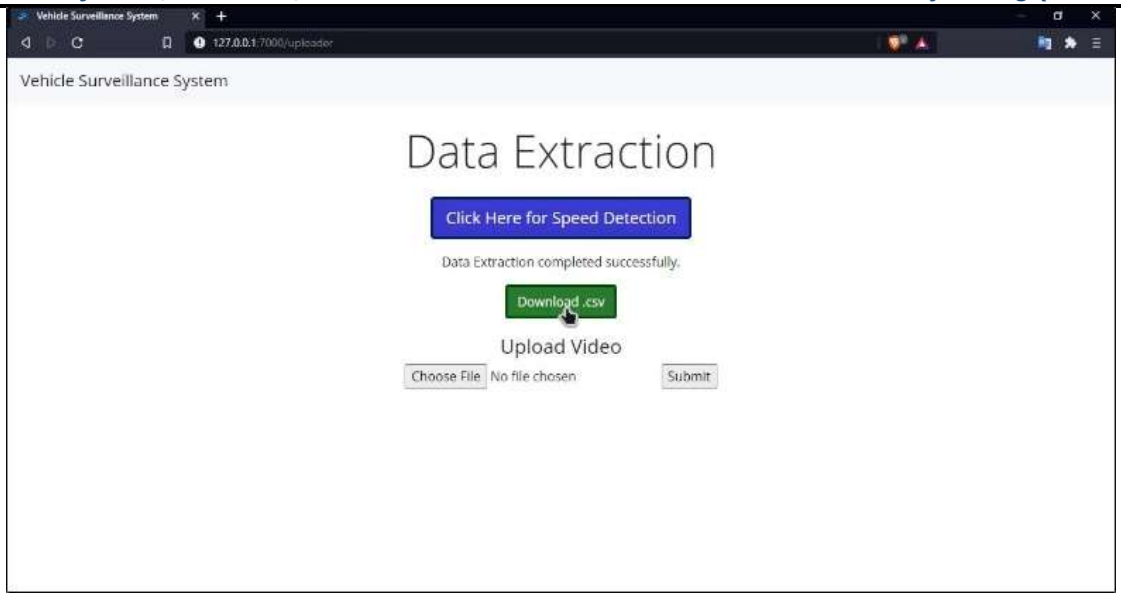

Figure 4.11 : Downloading Data Extraction output in a CSV File

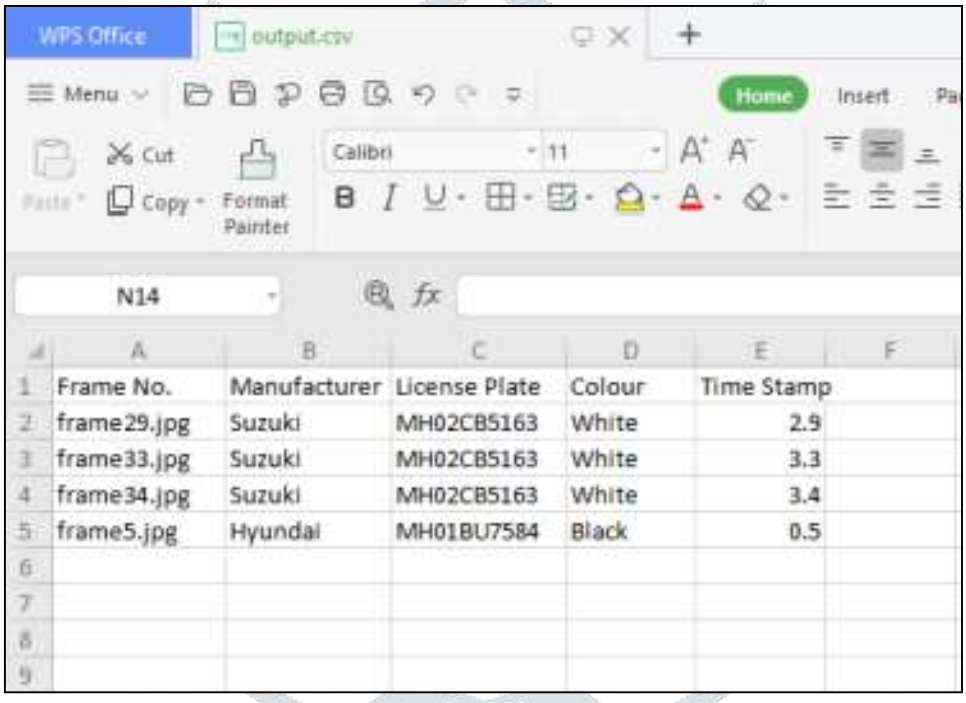

Figure 4.12: Final Data Extraction Output in CSV Format

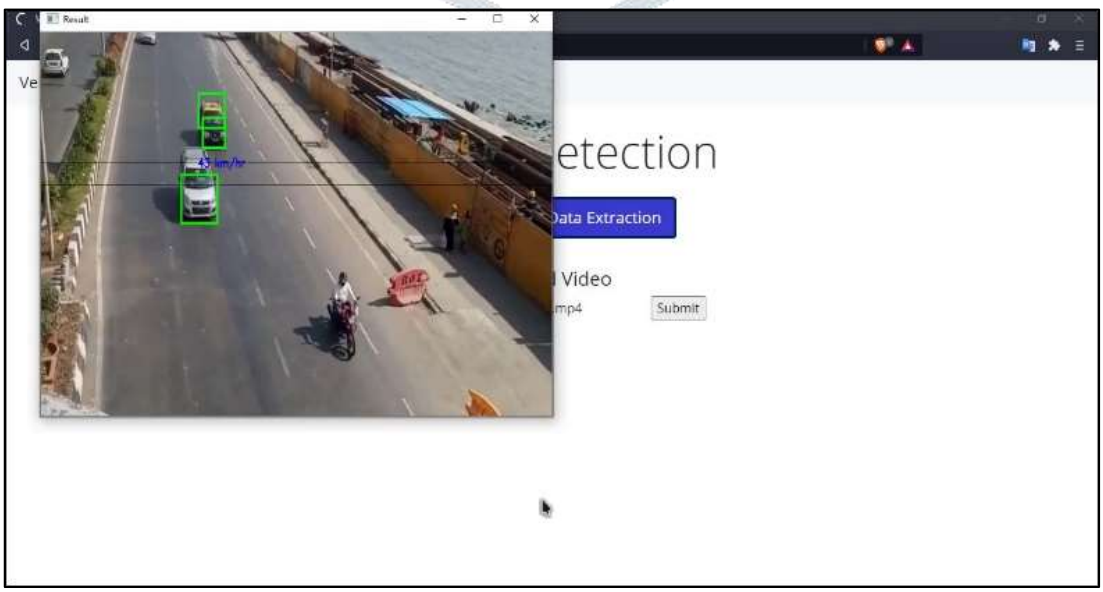

Figure 4.13 : Speed Calculation

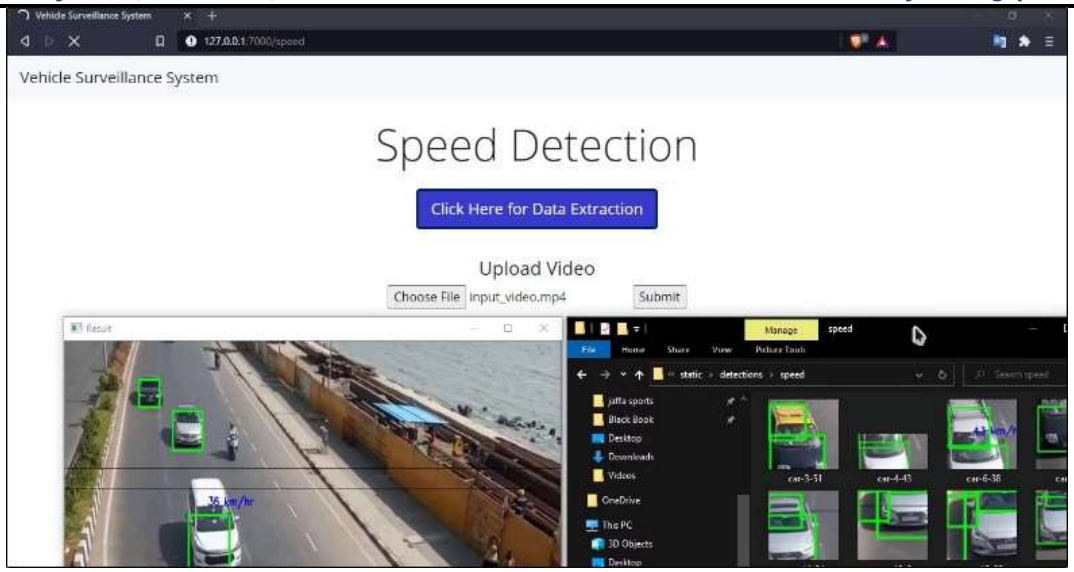

Figure 4.14: Storing into folders with ID and speed

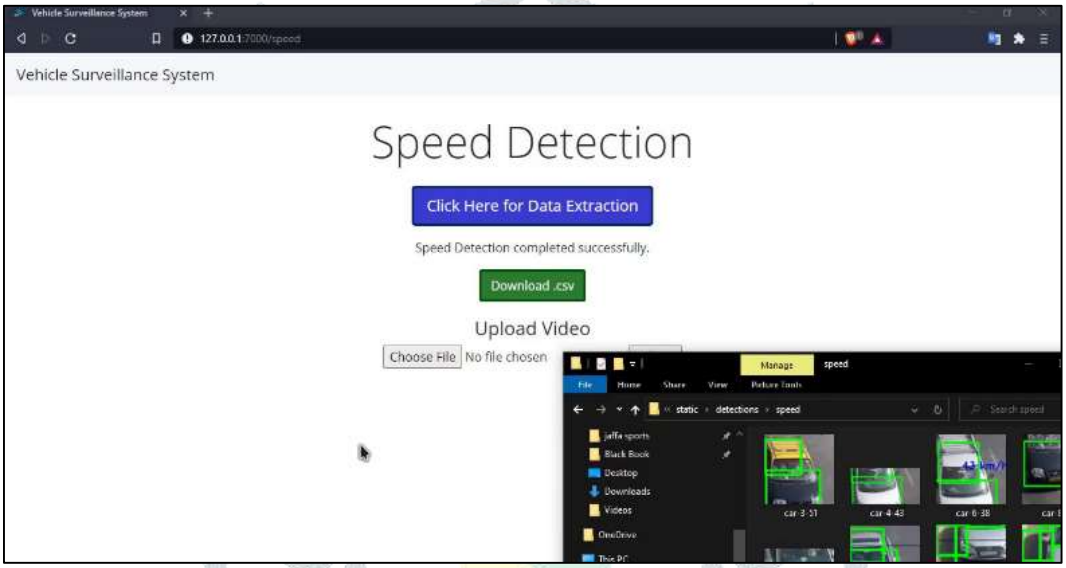

Figure 4.15: Downloading Speed Detection output in a CSV File

Car Id.	Speed	Timestamp
4	43	1.634718
6	38	1.534634
8	44	2.101781
3	51	1.901611
13	25	4.036754
12	32	4.303646
11	34	3.869946
16	36	4.403731
10	36	5.504664
17	39	6.105173
20	48	6.238619
22	47	6.905851
19	38	9.17444

Figure 4.16 : CSV file with ID, speed and timestamp

V. CONCLUSION

Our work will benefit the Intelligent Traffic System. It will provide them better and efficient way to analyse traffic. Time taken for data retrieval from the stored data is reduced. In case of videos now only images can be stored and we can analyse through images. Information in videos is converted to text files, which in turn minimizes storage space. Digital speed detection can help to reduce overspeed violations, which is the main cause for accidents.

VI. REFERENCES

[1] Vehicle Color Recognition on Urban Road by Feature Context Pan Chen, Xiang Bai, Member, IEEE, and Wenyu Liu, Member, IEEE.

[2] Vehicle Color Recognition using Convolutional Neural Network Reza Fuad Rachmadi and Ketut Eddy Purnama Department of Multimedia and Networking Engineering Institute Teknologi Sepuluh Nopember, Surabaya, Indonesia 60111 Object Detection CVPR 2019 paper.pdf

[3] https://www.kaggle.com/paulorzp/getting-a-car-color

[4] https://github.com/AlexeyAB/darknet

[5] https://wiki.loliot.net/docs/lang/python/libraries/yolov4/python-yolov4-about/

[6] The AI Guy - https://www.youtube.com/channel/UCrydcKaojc44XnuXrfhlV8Q

[7] keras-ocr - https://github.com/faustomorales/keras-ocr