



JAIN
DEEMED-TO-BE UNIVERSITY

FACULTY OF
ENGINEERING
AND TECHNOLOGY

Department of Information Science & Engineering

2021-2022

A Project Report on

**“SPEED DETECTION OF VEHICLE USING
MACHINE LEARNING ALGORITHM”**

Submitted in partial fulfilment for the award of the degree of

**BACHELOR OF TECHNOLOGY
IN
INFORMATION SCIENCE AND ENGINEERING**

Submitted by

**RISHAV SURANA
18BTRIS035**

**SUSANTA SARKAR
18BTRIS048**

**SIDDHARTH KUMAR HARLALKA
18BTRIS066**

**SUJATA BUDHA
18BTRIS065**

Under the guidance of

Mr. Mathiyalagan R,
Assistant Professor,
Department of Information Science & Engineering

Department of Information Science & Engineering

Jain Global campus, Kanakapura Taluk – 562112, Ramanagara District, Karnataka, India

CERTIFICATE

This is to certify that the project work titled **“SPEED DETECTION OF VEHICLE USING MACHINE LEARNING ALGORITHM”** is carried out by **Rishav Surana (18BTRIS035), Susanta Sarkar (18BTRIS048), Siddharth Kumar Harlalka (18BTRIS066), Sujata Budha (18BTRIS065)**, a bonafide students of Bachelor of Technology at the Faculty of Engineering & Technology, Jain University, Bangalore in partial fulfillment for the award of degree in Bachelor of Technology in Information Science & Engineering, during the year **2021-2022**.



Mr. Mathiyalagan R
Assistant Professor
Dept. of ISE,
Faculty of Engineering &
Technology,
Jain University
Date:

Dr. Arun N
Head Of Department,
Dept. of ISE,
Faculty of Engineering &
Technology,
Jain University
Date:

Dr. Hariprasad S A
Director,
Faculty of Engineering &
Technology,
Jain University
Date:

Name of the Examiner

Signature of Examiner

1.

2.

DECLARATION

We, **Rishav Surana (18BTRIS035), Susanta Sarkar (18BTRIS048), Siddharth Kumar Harlalka (18BTRIS066), Sujata Budha (18BTRIS065)**, are students of eighth semester B-Tech in **Information Science & Engineering**, at Faculty of Engineering & Technology, **Jain University**, hereby declare that the project titled **“Speed Detection of Vehicle Using Machine Learning Algorithm”** has been carried out by us and submitted in partial fulfilment for the award of degree in **Bachelor of Technology in Information Science & Engineering** during the academic year **2018-2022**. Further, the matter presented in the project has not been submitted previously by anybody for the award of any degree or any diploma to any other University, to the best of our knowledge and faith.

Name1: Rishav Surana
USN: 18BTRIS035

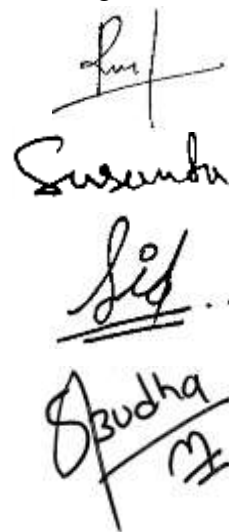
Name2: Susanta Sarkar
USN: 18BTRIS048

Name3: Siddharth Kumar Harlalka
USN: 18BTRIS066

Name4: Sujata Budha
USN: 18BTRIS065

Place: Bangalore
Date:

Signature



The block contains four handwritten signatures stacked vertically. The first signature is for Rishav Surana, the second for Susanta Sarkar, the third for Siddharth Kumar Harlalka, and the fourth for Sujata Budha. Each signature is written in black ink and is distinct from the others.

ACKNOWLEDGEMENT

It is a great pleasure for us to acknowledge the assistance and support of a large number of individuals who have been responsible for the successful completion of this project work.

First, we take this opportunity to express our sincere gratitude to Faculty of Engineering & Technology, Jain University for providing us with a great opportunity to pursue our Bachelor's Degree in this institution.

*In particular we would like to thank **Dr. Hariprasad S A, Director, Faculty of Engineering & Technology, Jain University** for his constant encouragement and expert advice.*

*It is a matter of immense pleasure to express our sincere thanks to **Dr. Arun N, Head of the department, Information Science & Engineering, Jain University,** for providing right academic guidance that made our task possible.*

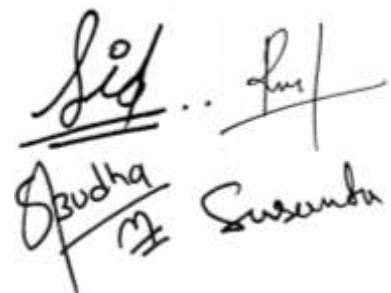
*We would like to thank our guide **Mr. Mathiyalagan R, Assistant Professor, Department of Information Science & Engineering, Jain University,** for sparing his valuable time to extend help in every step of our project work, which paved the way for smooth progress and fruitful culmination of the project.*

*We would like to thank our Project Coordinator **Prof. Soumya K N** and all the staff members of Information Science & Engineering for their support.*

We are also grateful to our family and friends who provided us with every requirement throughout the course.

We would like to thank one and all who directly or indirectly helped us in completing the Project work successfully.

Signature of Students



The block contains four handwritten signatures. The first signature is 'Sid', the second is 'Ansh', the third is 'Sudha', and the fourth is 'Sushanta'. The signatures are written in a cursive style.

ABSTRACT

The project is intending to develop a vehicle tracking and speed estimation and the object detection using machine learning and open cv algorithm. The System is designed to track the vehicle position and calculate its moving speed. It identifies the features of images that generates an intelligent understanding of images just like human vision works. We have gone through a brief introduce of machine learning and object detection framework like some of the approaches which we can use are YOLO, Haar Cascade, Decision tree etc. Object Detection is related to Computer Vision. Object detection enables detecting every instance of objects in images and videos. It identifies the feature of Images that generates an intelligent understanding of images just like human vision works. In this we begin with the brief introduction of machine learning and object detection framework like Region-Based Convolutional Neural Network (RCNN), You only look once (YOLO), Haar Cascade, Decision Tree etc. Then we focus on our proposed object detection architectures along with some modifications. The traditional model detects a certain object in images. Vehicle detection and tracking is one of the key components of the smart traffic concept. Modern city planning and development is not achievable without proper knowledge of existing traffic flows within the city. Surveillance video is an undervalued source of traffic information, which can be discovered by variety of information technology tools and solutions, including machine learning techniques. We have designed and developed an adaptive video-based vehicle detection, classification, counting, and speed-measurement tool using Python programming language and OpenCV for real-time traffic data collection. It can be used for traffic flow monitoring, planning, and controlling to manage transport network as part of implementing intelligent transport management system in smart cities. The proposed system can detect, classify, count, and measure the speed of vehicles that pass through on a particular road. It can extract traffic data in csv/xml format from real-time video and recorded video, and then send the data to the central data-server. The proposed system extracts image frames from video and apply a filter based on the user-defined threshold value. We have applied MOG2 background subtraction algorithm for subtracting background from the object, which separates foreground objects from the background in a sequence of image frames. The proposed system can detect, classify, and count vehicles of different types and size as a plug & play system. An efficient traffic management system is needed in all kinds of roads, such as off roads, highways, etc... Though several laws and speed controller has been attached to the vehicles, Speed limit may vary from road to road. Still Traffic management system faces different kinds of challenges every day and its being a research area though number of proposals has been identified. Many numbers of methods have been proposed in computer Vision and machine learning approaches for object tracking. In this paper vehicles are identified and detected using a video that taken from surveillance camera.

Table of Contents

CHAPTER 1	1
1.1 Introduction	2
CHAPTER 2	5
2.1 Literature Survey	6
2.1.1 Research paper 1	6
2.1.2 Research paper 2	7
2.1.3 Research paper 3	8
2.1.4 Research paper 4	9
2.1.5 Research paper 5	10
2.1.6 Research paper 6	11
2.1.7 Research paper 7	12
2.1.8 Research paper 8	13
2.1.9 Research paper 9	14
2.1.10 Research paper 10	14
CHAPTER 3	16
3.1 Basic Work	17
3.1.1 Existing System and Disadvantage	17
3.1.2 Proposed System and Advantages	17
3.2 Activity diagram	18
3.3 Limitations of the Current Work	19
3.4 Problem Definition	19
3.5 Objectives and methodology	19
3.5.1 Objectives	19
3.5.2 Methodology	20
3.6 The overview of system	21
CHAPTER 4	22
4.1 Tool Description	23
4.1.1 Software	23
4.1.2 Hardware	35
CHAPTER 5	36
5.1 RESULTS AND DISCUSSION	37
CHAPTER 6	41
6.1 CONCLUSIONS AND FUTURE SCOPE	42
6.1.1 Conclusions	42
6.1.2 Future Scope	42

REFERENCES	vii
APPENDIX-I	ix
8.1 SOURCE CODE	ix
8.1.1 Vech.xml	ix
8.1.2 Main.py	xxiii
INFORMATION REGARDING STUDENT	xxix
BATCH PHOTOGRAPH ALONG WITH GUIDE	xxx

Table of Figure

FIG. 3.3 ACTIVITY DIAGRAM.....	18
FIG. 3.1 VEHICLE IMAGE.....	21
FIG. 3.2 NON- VEHICLE IMAGE	21
FIG. 4.1 PYTHON LOGO.....	23
FIG. 4.2 SPYDER LOGO.....	25
FIG. 4.3 OPENCV LOGO.....	27
FIG. 4.4 DLIB LOGO	29
FIG. 4.5 VISUAL STUDIO CODE LOGO	30
FIG. 4.6 JUPYTER NOTEBOOK LOGO	31
FIG. 4.7 ANACONDA LOGO	33
FIG. 4.8 CASCADE TRAINER GUI LOGO.....	34
FIG. 4.1 GUI APP OF CASCADE TRAINER.....	37
FIG. 4.2 TRAINING PHASE.....	37
FIG. 4.3 TRAINING COMPLETED.....	38
FIG. 4.4 OBJECT DETECTION.....	38
FIG. 4.5 OBJECT TRACKING	39
FIG. 4.6 SPEED ESTIMATION.....	39
FIG. 4.7 RECORD OF OVER-SPEEDING VEHICLES.....	40

NOMENCLATURE

CV	Computer Vision
YOLO	You Only Look Once
PPM	Pixel Per Meter
HMM	Hidden Markov Model
GUI	Graphical User Interface
XML	Extensible Markup Language
ITS	Intelligent Traffic System

CHAPTER 1

Chapter 1

1.1 Introduction

Vehicle speed estimation is one of the most important concepts in traffic controlling systems. Since, speed limit violation is vital to control the roadways and prevent drivers from excessive speed. Speeding is one of the biggest traffic violations. It endangers everyone on the road. Some of the consequences of over speeding are loss of vehicle control, increased stopping distance, economic losses, increased fuel consumption and loss of lives. As crash speeds get very high, airbags and seat belts may not work as well to protect the passengers from the collision. Over speeding costs of billions of dollars to the country's economy.

Also, proper management of roadways decreases the dangerous accidents and makes the roads safer. One of the objectives is to identify the over speed vehicles and drivers are getting fined. Manual operations are being done everywhere using radar concepts to detect the over speed vehicles. However, with the fast-growing technology we can detect the over speed vehicles without manual intervention. Computer Vision and Machine learning technologies helps to automatically detect the over speed vehicles and drivers can be fined. Even though, various methods and techniques have been proposed to estimate the speed of vehicles, there still remain some limitations, and thus, studies on speed estimation still continue. Recently, digital cameras provide high quality images, which are even well-situated for speed estimation. Not only are digital cameras much less expensive but also, they needed simple installation and maintenance than the other similar gadgets. Hence considering the cost and maintenance, video cameras play a good role to detect the over speed vehicles.

Compared to conventional artificial observation alarming, this detection has advantages of lower manual intervention, fast reaction, high detection rate, treatment and responsibility identification for post-accident so it has become a popular research direction in intelligent traffic field. Video surveillance has attached more and more people's attentions. In this paper vehicles are identified using computer vision technique and machine learning algorithms, then object being tracked using dlibs functions.

The aim of this project is to design an automated over speed detection system that would notify traffic police with the details of an over speeding vehicle. Once the details are sent to the regulatory authority, the driver can be charged for over speeding. The main objective of the project is to eliminate the manpower needed by the existing systems. Currently, officer has to hold the speeding gun to measure the speed of the vehicles. It should be noted that currently, only 5% of the violators get speeding tickets. This is because the number of vehicles is much more compared to law enforcement officers.

Since the population and transport system increase day by day, the demand for managing them increase at the same time. The world is getting populated so fast. Therefore, the number of machines from any types including vehicles increased at the same time. That

being said, new topics like traffic, accidents and many more issues are needed to be managed. It is hard to manage them with the old methods, new trends and technologies have been found and invented to handle each and every milestone that human kind is trying achieve. One of these challenges is traffic in highways and cities. Many options like traffic light, sign, etc. deployed in order to deal with these phenomena. It seems that these options are not enough or not so efficient alone. New technologies like object detection and tracking are invented in order to utilize automated camera surveillance to produce data that can give meanings for a decision-making process. These phenomena have been used for different kind of issues. The new trend Intelligent Transport System (ITS) has many elements which object detection and tracking are one of them. This system is used to detect vehicles, lanes, traffic sign, or vehicle make detection. The vehicle detection and classify ability gives us the possibility to improve the traffic flows and roads, prevent accidents, and registering traffic crimes and violations.

Humans can easily recognize vehicles in videos or images or to identify different types of cars. In computer algorithms and programs, it is highly depended on the types of data. Some challenges like the weather or light are also plays important role on making the process easy or much hard. At the same time, we have different types and shapes of vehicles. More than that the new challenge could be to identify moving objects in a video in real time where they are different in size and shape. There are different techniques and methods for vehicle detection and classification. The variety of these techniques are in types of 2 algorithms like Support Vector Machine (SVM), Convolutional Neural Network (CNN), Decision Tree, Recurrent Neural Network (RNN) etc. The field is constantly evolving since the industry is focused on this system or Computer visionary. In this thesis we investigate two algorithms SVM and Decision Tree to identify how they can apply in the field and which one works better than the other.

In the recent years we can see there is a vast increase in the number of vehicles all around the globe. Along with the increase in number of vehicles increases the number of accidents. Therefore, it is important to limit the speed of the vehicles at certain zones or areas. Radar speed measurement tools are commonly used for this purpose which can be inaccurate in certain cases such as in sensing smaller vehicles with weaker echoes. Also, it is difficult for these tools to detect vehicles changing in speeds too often or fast. Therefore, there is a need for a better technique to detect the speed of the moving vehicles. Then using expensive sensors such as radars, the vehicles video streaming could be used for this purpose. The video stream of the moving vehicle is given as an input, then it is passed through the filter for detecting its speed.

Determining vehicle speed is an important task for urban traffic surveillance. The information may be used not only to issue fines when drivers exceed speed limits, but also to feed systems such as traffic controllers. Vehicle speed can be measured by intrusive or nonintrusive technologies. Intrusive systems, usually based on inductive loop detectors, are highly sensible and accurate, but have high installation costs and can damage the roadway. Non-intrusive systems are mostly based on laser sensors, infrared sensors, Doppler radar, or

audio-based sensors [1, 2]. They can be portable, but are expensive and require frequent maintenance. In many cities, inductive loop detectors are installed along with cameras, which are used to identify the license plates of vehicles that exceed the speed limit. If the speed information can be extracted from the already available image data, we may obtain a non-intrusive system with significantly reduced costs. In this paper, we describe a novel system for estimating vehicle speed from videos captured in urban roadways.

Thanks to the improvement that is achieved in computer vision and machine learning, we can find application of these methods in many other areas. One of them is traffic monitoring and management system, where the importance is still growing with growing urbanization. This paper aims at speed detection or estimation of vehicles from video stream. Nowadays the most common way to measure speed is by using the radar equipment, therefore it is very important to propose any other concepts like measuring vehicle speed from video stream. Instead of hardware dependency that is problem with radar systems we can use image processing, which is mainly based on software implementation.

Computer vision technology is using for traffic monitoring in many countries [10], [11]. The development of computer vision technology over video-based traffic monitoring for detecting moving vehicles in video streams become an essential part in ITS [12], [13]. A good number of works has been done on vehicle tracking and detection using computer vision technology. In 2005, Hasegawa and Kaneda [14] introduced a system for detecting and classifying the moving objects by its type and color. In this process, a series of images of a specific location were supplied and vehicles from these images were identified. In 2013, Nilesh et al. [15] designed and developed a system using visual C++ with OpenCV for detecting and counting moving vehicles. It can automatically identify and count moving objects as vehicle in real-time or from recorded videos, which basically used background subtraction, image filtering, image binary and segmentation method. In 2014, Da Li et al. [16] developed real-time moving vehicle detection, tracking, and counting system also using Visual C++ with OpenCV including adaptive subtracted background method in combination with virtual detector and blob tracking technology. Virtual detector constructs a set of rectangular regions in each input image frame and blob tracking method generates input image frames, the absolute difference between the background image and foreground blobs corresponding to the vehicles on the road. The above systems have some limitations like tackling shadows, occlusion of multiple vehicles that appear in a single region.

CHAPTER 2

Chapter 2

2.1 Literature Survey

2.1.1 Research paper 1

[1] Friedman. N and Russell. S. in their paper “Image Segmentation in Video Sequences.” have described the detail of method in 2 sections:

- Background subtraction: The roots of background subtraction go back to the 19th century, when it was shown that the background image could be obtained simply by exposing a film for a period of time much longer than the time required for moving objects to traverse the field of view.
- Pixel model: Consider a single pixel and the distribution of its values over time. Some of the time it will be in its “normal” background state—for example, a small area of the road surface. Some of the time it may be in the shadow of moving vehicles, and some of the time it may be part of a vehicle. Thus, in the case of traffic surveillance, we can think of the distribution of values i_x, y of a pixel (x, y) as the weighted sum of three distributions r_x, y (road), s_x, y (shadow), and v_x, y (vehicle): $i_x, y = w_x, y (r_x, y, s_x, y, v_x, y)$.

One can expect much better performance using more background knowledge encoded as probabilistic models. Heuristic approach may not work well in extremes of lighting conditions.

"Background subtraction" is an old technique for finding moving objects in a video sequence for example, cars driving on a freeway. The idea is that subtracting the current image from a time averaged background image will leave only nonstationary objects. It is, however, a crude approximation to the task of classifying each pixel of the current image; it fails with slow-moving objects and does not distinguish shadows from moving objects. The basic idea of this paper is that we can classify each pixel using a model of how that pixel looks when it is part of different classes. We learn a mixture-of-Gaussians classification model for each pixel using an unsupervised technique- an efficient, incremental version of EM. Unlike the standard image-averaging approach, this automatically updates the mixture component for each class according to likelihood of membership; hence slow-moving objects are handled perfectly. Our approach also identifies and eliminates shadows much more effectively than other techniques such as thresholding. Application of this method as part of the Road watch traffic surveillance project is expected to result in significant improvements in vehicle identification and tracking.

2.1.2 Research paper 2

[2] Margrit Betke, Esin Haritaoglu and Larry S. Davis in their paper “Real-time multiple vehicle detection and tracking from a moving vehicle.” have given idea about the smart camera assisted car that can interpret its surroundings automatically, robustly, and in real time.

The methods used by authors are:

- The hard real-time system: The goal of their vision system is to provide a car control system with a sufficient analysis of its changing environment, so that it can act to the dangerous situation straight away.
- Vehicle detection and tracking: The input data of the vision system consists of image sequences taken from a camera mounted inside our car, just behind the windshield. The images show the environment in front of the car the road, other cars, bridges, and trees next to the road. The primary task of the system is to distinguish the cars from other stationary and moving objects in the images and recognize them as cars. This is a challenging task, because the continuously changing landscape along the road and the various lighting conditions that depend on the time of day and weather are not known in advance. Recognition of vehicles that suddenly enter the scene is difficult.

The results demonstrate robust, real-time, car recognition and tracking over hundreds and thousands of image frames. At night on city expressways, when there are many city lights in the background, the system has problems finding vehicle outlines and distinguishing vehicles on the road from obstacles in the background. The general goal of our research is to develop an intelligent, camera-assisted car that is able to interpret its surroundings automatically, robustly, and in real time. Even in the specific case of a highway’s well-structured environment, this is a difficult problem. Traffic volume, driver behavior, lighting, and road conditions are difficult to predict. Our system therefore analyzes the whole highway scene. It segments the road surface from the image using color classification, and then recognizes and tracks lane markings, road boundaries and multiple vehicles. We analyze the tracking performance of our system using more than 1 h of video taken on American and German highways and city expressways during the day and at night. The city expressway data includes tunnel sequences. Our vision system does not need any initialization by a human operator, but recognizes the cars it tracks automatically. The video data is processed in real time without any specialized hardware. All we need is an ordinary video camera and a low-cost PC with an image capture board.

Due to safety concerns, camera-assisted or vision-guided vehicles must react to dangerous situations immediately. Not only must the supporting vision system do its processing extremely fast, i.e., in soft real time, but it must also guarantee to react within a fixed time frame under all circumstances, i.e., in hard real time. A hard real-time system can predict in advance how long its computations will take. We have therefore utilized the advantages of the hard real-time operating system called “Maruti,” whose scheduling guarantees – prior to any execution – that the required deadlines are met and the vision system will react in a timely manner [56]. The soft real-time version of our system runs on the Windows NT 4.0 operating system.

2.1.3 Research paper 3

[3] Seda Kul, Süleyman Eken, Ahmet Saya in their paper “Distributed and collaborative real-time vehicle detection and classification over the video streams.” have given these steps:

- Background subtraction and detection of vehicles Region of Interest (ROI).
- Extraction of features representing vehicles.
- Dimensionality reduction to reduce dependencies of some features.
- Vehicle classification.
- Designing a distributed system for filtering and transferring classified images to the registered user.

Traffic surveillance cameras are widely used in traffic management and information systems. There are some studies involving image and video processing on traffic surveillance data, mostly used in traffic management systems. Some of these studies are as follows: automatic vehicle identification,¹ road capacity,² traffic density measurement,³ speed detection,^{4–6} traffic violation detection,⁷ and specifying car categories.^{8–11} Vehicle classification is a very important process in traffic management systems. Outcomes of the vehicle classification can be used for different aims such as reporting and statistical analysis, determining the asphalt thickness, and searching for vehicles for specific classes which are built by classifying their varying sizes, colors, velocities, and so on. Advancement in real-time image processing over the Internet draw attention in many application areas. In intelligent traffic/vehicle management systems, transferring traffic surveillance video data from source to destination for further processing is very important. Surveillance cameras have limited processing power and small storage area. Therefore, they can only capture and store data locally and perform some basic image processing. Some even do not have storage or processing capabilities. They only capture data and transfer to the processing units via Internet connections. However, in some traffic emergency systems, real-time image processing on temporal image sequences is required. Since video data are a streaming media and get very large in size, the central processing techniques are not efficient in their processing. It even gets worse in the case of real-time scenarios. Moreover, in such systems, they need to be processed and analyzed in a reasonable time period. For the decision making, the right data need to be dispatched to the right places in right time.

In real world, a user may want to get an image or a piece of video of small-sized vehicles (such as sedan), another user may want to get the same information about largesized vehicles (such as trucks), and one another user may want to get the same video streaming about a vehicle travelling in a wrong lane and wrong direction. To realize such a system, a surveillance camera, a dispatcher, and end users are needed. The surveillance camera transfers video streams to the dispatcher. The dispatcher processes the video stream and chunks it into pieces according to the registered end users and then transfers the related video streams to the registered end users.

2.1.4 Research paper 4

[4] Jozef Gerát, Dominik Sopiak, Miloš Oravec, Jarmila Pavlovicová in their paper “Vehicle Speed Detection from Camera Stream Using Image Processing Methods.” have used various method that are described below: Gaussian mixture model: It is a common method for real-time segmentation of moving regions in image sequences. Each pixel is classified as foreground or background based on his representation in the most suitable Gaussian distribution. DBSCAN clustering: Method relies on a density-based notion of clusters which is designed to discover clusters of arbitrary shape. Given a set of foreground points from Gaussian mixture model, it groups together points that are closely packed together (points with many nearby neighbors), marking as outliers points that lie alone in low-density regions (whose nearest neighbors are too far away). With the use of optical flow and kalman filter methods, it helped to solve the problem with overlays with static foreground and also improves speed detection. There is errors on boundaries of moving object.

Thanks to the improvement that is achieved in computer vision and machine learning, we can find application of these methods in many other areas. One of them is traffic monitoring and management system, where the importance is still growing with growing urbanization. This paper aims at speed detection or estimation of vehicles from video stream. Nowadays the most common way to measure speed is by using the radar equipment, therefore it is very important to propose any other concepts like measuring vehicle speed from video stream. Instead of hardware dependency that is problem with radar systems we can use image processing, which is mainly based on software implementation. There are several papers where many vehicles speed measuring techniques. Most important part of speed measurement system is object detection and tracking are proposed. Where frame differencing is used for moving object detection. That method is very efficient in case, where is no movement in the background. More suitable way for object detection. Gaussian mixture model is presented as solution for foreground subtraction with dynamic background. For object tracking optical flow is used that helps to determine not only speed of movement but also direction of the motion.

Very interesting project is mentioned in [5], where vehicle speed detection system runs on embedded device Raspberry Pi. Object detection is solved also by Gaussian Mixture model, which proves that this method is suitable also for devices with limited resources. Speed estimation from video frames using corner detection is shown in [6]. For vehicle segmentation also combination of edge detectors, corner detection and morphological operations is used. Another way of using image processing technique to detect speed is shown in [7]. Their goal is to detect speed from single blurred image.

2.1.5 Research paper 5

[5] Xiao-jun Tan, Jun Li & Chunlu Liu in their paper “A video-based real-time vehicle detection method by classified background learning.” have described Video-based vehicle detection has played an important role in real-time traffic management systems over the past decade. Video-based traffic monitoring systems offer a number of advantages over traditional methods, such as loop detectors. In addition to vehicle counting, more traffic information can be obtained by video images, including vehicle classifications, lane changes, etc. Furthermore, video cameras can be easily installed and used in mobile environments.

Real-time vehicle detection in a video stream relies heavily on image processing techniques, such as motion segmentation, edge detection and digital filtering, etc. Recently, different vision-based vehicle detection approaches have been proposed by researchers. However, two main concerns arise when they are applied to real-time vehicle detection, namely their robustness and performance. The methods should be robust enough to fit in tough outdoor environments; the algorithms should also be efficient so that the detection can be finished in time. The system can be self-adaptive to variant illumination and tolerate small motion changes caused by wind or other factors. The proposed method is not suitable for vehicle detection at night when there is insufficient illumination. Video-based vehicle detection has played an important role in real-time traffic management systems over the past decade. Video-based traffic monitoring systems offer a number of advantages over traditional methods, such as loop detectors. In addition to vehicle counting, more traffic information can be obtained by video images, including vehicle classifications, lane changes, etc. Furthermore, video cameras can be easily installed and used in mobile environments.

Real-time vehicle detection in a video stream relies heavily on image processing techniques, such as motion segmentation, edge detection and digital filtering, etc. Recently, different vision-based vehicle detection approaches have been proposed by researchers [1-5]. However, two main concerns arise when they are applied to real-time vehicle detection, namely their robustness and performance. The methods should be robust enough to fit in tough outdoor environments; the algorithms should also be efficient so that the detection can be finished in time.

Recent applications of high-resolution cameras have yielded much higher performance requirements. In this article, the authors present a two-level method for real-time vehicle detection that achieves the following two goals:

- More robust: the system can be self-adaptive to variant illumination and tolerate small motion changes caused by wind or other factors.
- High performance: the algorithm requires low CPU usage so that vehicles can be detected in the required time.

The article is organised as follows: a review of background subtraction is first given and followed by a proposed two-level method. Several scene tests are used in order to compare the proposed method with previous methods. There is also a discussion given in the last section.

2.1.6 Research paper 6

[6] Diogo C. Luvizon, Bogdan T. Nassu and Rodrigo Minetto in their paper “Vehicle speed estimation by license plate detection and tracking.” they describe a novel system for estimating vehicle speed from videos captured in urban roadways. The system’s pipeline is First, a text detector is used to detect the license plates of passing vehicles. Stable features inside the detected regions are then tracked, using a combination of the SIFT and KLT algorithms. After filtering out inconsistencies, the vehicle speed is estimated by comparing feature trajectories to known real world measures, which allow us to rectify the perspective distortion and obtain a meter-per-pixel relation. To our knowledge, our system is the first to estimate vehicle speed by tracking corner and region features extracted from the license plate.

Vehicle speed can be measured by intrusive or non-intrusive technologies. Intrusive systems, usually based on inductive loop detectors, are highly sensible and accurate, but have high installation costs and can damage the roadway. Non-intrusive systems are mostly based on laser sensors, infrared sensors, Doppler radar, or audio-based sensors. They can be portable, but are expensive and require frequent maintenance. In many cities, inductive loop detectors are installed along with cameras, which are used to identify the license plates of vehicles that exceed the speed limit. If the speed information can be extracted from the already available image data, we may obtain a non-intrusive system with significantly reduced costs. The system can be self-adaptive to variant illumination and tolerate small motion changes caused by wind or other factors. Extracts data only when camera is closer to vehicle, errors on boundaries of moving object.

Determining vehicle speed is an important task for urban traffic surveillance. The information may be used not only to issue fines when drivers exceed speed limits, but also to feed systems such as traffic controllers. Vehicle speed can be measured by intrusive or nonintrusive technologies. Intrusive systems, usually based on inductive loop detectors, are highly sensible and accurate, but have high installation costs and can damage the roadway. Non-intrusive systems are mostly based on laser sensors, infrared sensors, Doppler radar, or audio-based sensors [1, 2]. They can be portable, but are expensive and require frequent maintenance. In many cities, inductive loop detectors are installed along with cameras, which are used to identify the license plates of vehicles that exceed the speed limit. If the speed information can be extracted from the already available image data, we may obtain a non-intrusive system with significantly reduced costs.

2.1.7 Research paper 7

[7] Kiran Kumar KV, Pallavi Chandrakant, Santosh Kumar, Kushal KJ in their paper “Vehicle Speed Detection in Video frames using Corner detection.” Have describe the increase in urban population in many cities, amounts of vehicles have also been drastically increased. In a recent study over-speeding caused most of the accidents, followed by drunken driving. Over-speeding of two-wheelers and three-wheelers is one of the major reasons of accidents. In order to support traffic management system in our country we need to build economical traffic monitoring systems. In recent times image and video processing has been applied to the field of traffic management system. This paper explicitly concentrates on the speed of the vehicles, which is one of the important parameters to make roads safe. Relatively few efforts have been attempted to measure speed by using video images from uncalibrated cameras. Similarly, several other papers suggest estimating speed by first placing two detection lines (separated by a known distance) and then measuring travel times between the lines. This paper provides a low cost and versatile vehicle speed detection using a computer vision-based approach. In this setting, the speed is detected using video cameras commonly available. Distance and angle covered by this paper is more when compared to radar gun. Efficiency goes down to multiple vehicles. Focus should be on a single vehicle.

The speed detection system is able detect vehicles speed even with shadows also. With fast processors or high-end smart phones, it can be seen as a future vehicles speed detector. Since the cost of this system is many times less. It can used manage traffic and avoid accidents at a cheaper price. Project gives the correct result under normal conditions and cloudy conditions. But fails if shadows creep up during sunset and sunrise.

One of the technologies our law enforcement department uses to measure the speed of a moving vehicle is Doppler radar. It beams a radio wave at a vehicle, and then estimate the vehicles speed by measuring change in reflected wave frequency. It is a fixed or hand-held device and is reliable when a moving object is in the field of view and no other moving objects are nearby. Cosine error has to be taken care if the gun is not in the line of sight. Also Radio interference which causes errors in speed detection has to be taken care. Some of the previous works using image and video processing applied for vehicle detection and speed measurements are vehicle detection based on frame difference, calibrated camera, motion trajectories, Optics and digital aerial images. Also, blurred images were used to find out the vehicle speed along with high-end camera motion detection for automated speed measurements and feature point tracking for vehicle speed measurements were used. Currently highly reliable GPS systems are used to track vehicle speeds in US. Cost-effectiveness is a concern in such a case. In our method moving vehicle video from any video camera or mobile source is utilized. The algorithms are implemented in ‘C’ language using OpenCV and Visual Studio. Later this code can be ported to a simple processor where vehicle speed can be measured. Example: a simple smart phone with average processing capacity. Our aim was to implement real- time vehicle speed detector.

2.1.8 Research paper 8

[8] Denis Kleyko, Roland Hostettler, Wolfgang Birk, and Evofgeny Osipov in their paper “Vehicle Classification using Road Side Sensors and Feature free Data Smashing Approach.” the author has applied feature free data smashing method to the problem of vehicle classification on magnetometer and accelerometer measurements from road side sensors. Data Smashing Approach: It is a data mining approach which is suitable for comparison of two arbitrary data streams with each other. It does not require the extraction of features from raw signals and it was evaluated on a large dataset. Main drawback of data smashing usage is its moderate performance (76.8% vs. 93.4% for the state-of-the-art feature-based method) shown for largest data.

This method was used to solve two major problems: clustering (unsupervised learning) and classification (supervised learning). For this problems data smashing showed high performance. Vehicle classification is an important task in traffic monitoring and analysis. Rich information about the traffic composition provided by a classification analysis is commonly used for different purposes such as urban planning, road maintenance, traffic light scheduling, etc. For a long time, this kind of information has been obtained based on inductive loop detectors for permanent installations or pressure tubes for temporary installations. During the last decade, with the advance of cheap sensor technology, wireless communication, and electronics, sensor networks have started to replace those traditional systems. Some advantages of these novel approaches include the possibility of on-demand or real-time access to the data, and slower wear-off rate due to the possibility of non-invasive installations. Traffic monitoring using wireless sensor nodes as a such is a rather mature field. It was shown that magnetometers can be used to count traffic, to estimate vehicles’ speed, and even to classify vehicles. Furthermore, in it was shown how to use a road-surface mounted micro accelerometer and a neural networks-based algorithm to distinguish between diesel, gasoline, and heavy diesel engine vehicles using the frequency spectrum generated by a passing vehicle. The works in also used accelerometer-based vehicle detection. The authors developed a peak detection algorithm to detect individual vehicle axles followed by a table lookup. Authors in proposed the bio-inspired classifier using vibration measurements from the road side sensor.

The advantage of this classification approach is that it, in contrast to the most of traditional machine learning algorithms, does not require the extraction of features from raw signals. Extraction of relevant features from raw data is an engineering art on its own and usually requires a domain expert who understands the problem at hand. Moreover, the choice of features does significantly affect the classification performance. In contrast, when applying data smashing the requirements for the expert knowledge are minimized which positively affects development cost and time.

2.1.9 Research paper 9

[9] Jin-xiang Wang in his paper “Vehicle speed detection algorithm in video surveillance.” have explained vehicle detection is the most basic and the most important part of the intelligent transportation. Traditional detection technologies included in this study are infrared detection, the induction line detection, ultrasonic detection, acoustic array detection, radar, video image detection systems and so on. During the last decades, there are various new vehicle detection techniques in video surveillance are discovered. Vehicle detection technology based on video can obtain abundant information such as the vehicle speed, vehicle flow, vehicle type and so on from video image sequences, which has low cost and high efficiency. Vehicle motion analysis is a hot research field of computer vision. At present, the research of vehicle detection is mainly based on moving objects detection in video. Moving target detection is to extract the change area from the image sequence. The main methods for moving target detection are background difference method, frame difference method and optical flow method. The paper studied the vehicle speed detection algorithm based on moving target detection in video. The main work includes the following aspects: motion detection, vehicle tracking and vehicle speed estimation. The method mentioned in this paper has good robustness and strong practicability. Unable to get complete boundary information of moving object, sensitive to changes of the light, weather and other environmental conditions.

Background difference method subtracts the current real-time image frame to be detected and the original background image. The changed region will be determined as moving target area. At present, common background image estimation methods are statistical average method, median method and Surendra background updating algorithm and so on. The background difference method can obtain complete and accurate information of the moving target region. The deficiencies of the algorithm is that it is sensitive to changes of the light, weather and other environmental conditions. Optical flow method is based on constraint conditions of gray gradient unchanged and brightness preservation to detect moving target. Optical flow method can detect independently moving target without knowing any scene information in advance. But its calculation is too complex and time-consuming to achieve real-time detection.

2.1.10 Research paper 10

[10] R. Fraile and S. J. Maybank in their paper “Vehicle Trajectory Approximation and Classification.” have discussed the density of road traffic increases it becomes ever more important to detect quickly accidents or other abnormal events, both to save lives and to reduce the disruptive effects on traffic flow. Certain events can already be detected automatically using the image sequences obtained by fixed surveillance cameras which already line many motorways and main roads. These events include build ups in traffic density and vehicles coming to a halt. It is sufficient to use 2D or ‘blob based’ tracking provided the camera is placed high enough to avoid occlusions. 2D tracking is not sufficient to recover the more detailed information needed for accurate monitoring and control of road

traffic. This information can only be obtained by taking into account the 3D nature of the scene. For example, the trackers described in obtain a time sequence of measurements of the 3D position of a vehicle from a monocular image sequence by matching a wire frame model of the vehicle to the images. Once a sequence of measurements is obtained, the trajectory of the vehicle can be classified. For this we use a hidden Markov model (HMM).

HMMs can be the basis of fast and reliable algorithms for classifying trajectories and for identifying ones arising from abnormal driver behaviour. The trajectory classification method has only been applied to a limited number of trajectories.

We have described a method for finding low curvature approximations to segments of the trajectory of a car and then using quantised data from the sequence of approximations and a hidden Markov model (HMM) to classify the entire trajectory. The low curvature approximations are obtained only for short segments, but the approximation method can be extended to longer and more complicated trajectories. The trajectory classification method has only been applied to a limited number of trajectories. The results suggest that HMMs can be the basis of fast and reliable algorithms for classifying trajectories and for identifying ones arising from abnormal driver behaviour. Abnormal behaviour can be learnt by the HMM in the sense that the probability of an unusual sequence of states is low. Further work will involve extending the methods to a wider variety of vehicles and road situations. Integration of this method in a full decision system, taking into account site-dependent knowledge, such as the information given by visual traps, would have commercial applications.

CHAPTER 3

Chapter 3

3.1 Basic Work

3.1.1 Existing System and Disadvantage

A Existing system

Vehicle speed estimation is one of the most important concepts in traffic controlling systems. Since, speed limit violation is vital to control the roadways and prevent drivers from excessive speed as well as unwanted accidents.

Existing data collection methods such as radar sensor or inductive loop detector are slightly expensive and surveillance cameras are only used for manual check now. Turning low-cost cameras into effective sensors is a beneficial challenge. With computer vision and Machine learning techniques rapidly developed, this study tries to extract vehicle speeds from surveillance video data.

Optical flow is a technique used to describe image motion. The main problem with optical flow methods is that the smoothness of their motion does not allow discontinuities of motion across object boundaries in the scene. The assumption of constant flow (pure translation) for all pixels in a larger window is unreasonable for long periods of time.

Hidden Markov Model (HMM) is Viterbi (dynamic programming) algorithm which is expensive, both in terms of memory and compute time. For a sequence of length n , the dynamic programming for finding the best path through a model with s states and e edges takes memory proportional to sn and time proportional to end .

B Disadvantage

Although there are many ways to detect the objects but there are limitations. Multiple objects tracking at the same time is a difficult task to implement OR the efficiency of the model goes down when there are multiple objects in a frame or scene. Calculating the speed of vehicle using Kalman filter has limitation it has a slow reaction speed where there will be chance to miss the vehicle or skip the vehicle.

Working with Lucas-Kanade optical flow algorithm. The disadvantage of using this algorithm is that it can handle only a single vehicle at a time. Moreover, they require a side view of the vehicles and do not take perspective into account. Equation of spherical projection is used in to estimate the vehicle speed. Lucas-Kanade-Tomasi algorithm is used for motion tracking. Vehicle detection and tracking is one of the most important steps to estimate vehicle speed.

3.1.2 Proposed System and Advantages

A Proposed System

With consideration of existing system, the proposed idea of our project is implemented using

Haar cascade for object detection, Dlibs for object tracking and Euclidean distance for speed calculation. Training in the is done using cascade trainer GUI (Graphical User Interface). System is given an input as video which is captured using opencv and further it detects, track and estimate the speed of the vehicle. If the vehicle is found to be over-speeded then the system clicks the picture of a vehicle at that instance and this can be used for future purpose.

B Advantages

- It captures the images of the vehicle when over-speeding and will be used for further investigation.
- It reduces the human error as well as reduce the manpower.
- Detects multiple vehicles at same time.

3.2 Activity diagram

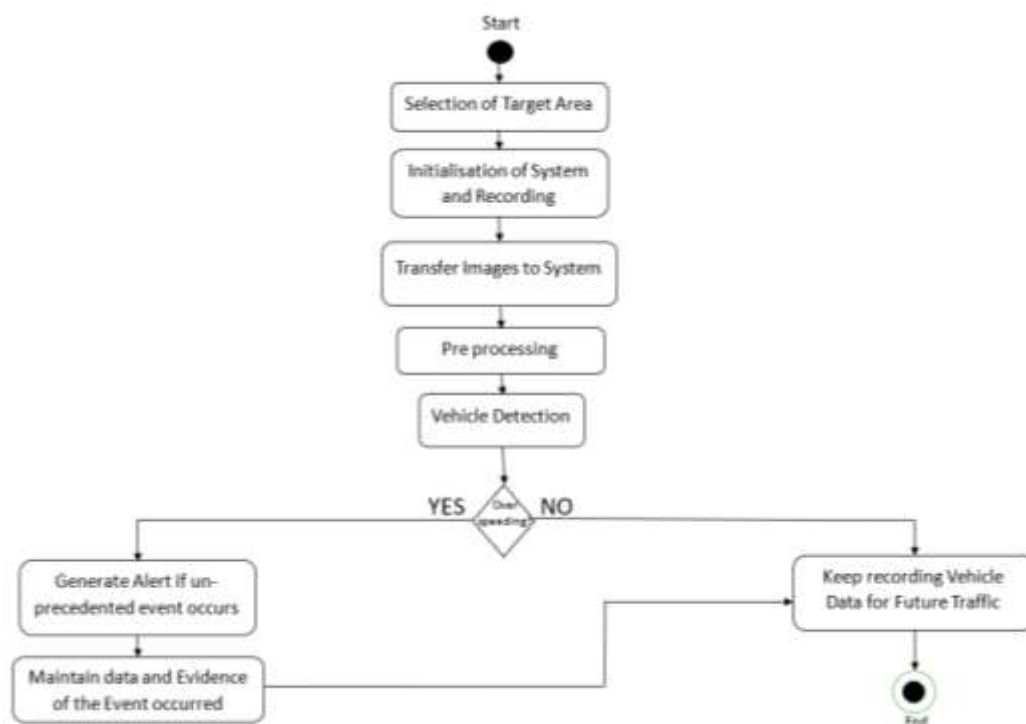


FIG. 3.3 ACTIVITY DIAGRAM

Vehicle Identification or detection refers the ability of computer and software system to recognize the object in an image or scene and recognize the object. Here we have built speed detection of vehicle using machine learning algorithm and image processing techniques. We have a dataset of images containing vehicles and non-vehicles. Vehicle images are called positive images or true images which contains the images of the vehicle taken from different angle. To increase the efficiency of the algorithm we use image processing techniques such as background subtraction and shadow removal. Non-vehicle images are called negative images or false images which contains the images of road, traffic sign, bridges and trees. We use haar cascade classifier Graphical User Interface (GUI) to train the images. The vehicle

detection is done using haar cascade trained file and the input is taken as a video file which can be taken from YouTube and any CCTV surveillance camera.

3.3 Limitations of the Current Work

- **Problem Formulation:** The over speeding vehicles detection problem is one of the fastest uprising problems in the country and to detect vehicle and its speed using CCTV is in research phase. A formal definition and formulation of the problem is required and necessary before studying the problem.
- **Calibration:** Each time we use new video from different CCTV then the program is need to be calibrated according to the video. Not only with respected to video but also with respect to different hardware and different system.
- **Visibility issues:** The program is totally dependent on video input as if the visibility gets poor then the detection and speed estimation of the vehicle get affected directly. Also, the night video of the CCTV can not be used for this program.
- **Hardware dependent:** This program is totally dependent on the hardware specification. As if the processor is fast then it might give different output as compared to slower processor.
- **Dataset Issue:** The current models have Dataset which has been trained with minimum pictures. This also affect the detection program as these datasets is used for detecting vehicle in the input video.

3.4 Problem Definition

The task of detecting and classifying objects in images and videos are classification task. To identify and detect vehicle in a video can be challenging as program should be trained in such a way that it can differentiate between vehicle and the background. Due to this reason background subtraction is compulsory. The reason behind this is to eliminate the load on processor for fast and accurate result. Task is suited well in machine learning since the task is a perform by CV.

3.5 Objectives and methodology

3.5.1 Objectives

Now days with rapid population growth it comes with heavy traffic. To manually check with radar system, it requires more man power which is also not efficient to manage heavy and control heavy traffic. Also, beside that all day being in between air pollution, they can suffer with sever health issue. Our system prevents that hard-working man with sever health issue and also detect each and every vehicle without radar machine and take a necessary action.

Every life is priceless and risking the one's life and others on the road is a punishable crime. To prevent that from happening this system will constantly monitor the flow of traffic

and record the details of the over speeding vehicle.

Smart traffic cam is still in developing stage and there are still many challenging issues that need further investigation. The potential of this research can help in self-driving cars.

3.5.2 Methodology

Here we have built speed detection of vehicle using machine learning algorithm and image processing techniques. We have a dataset of images containing vehicles and non-vehicles. Vehicle images are called positive images or true images which contains the images of the vehicle taken from different angle. Non-vehicle images are called negative images or false images which contains the images of road, traffic sign, bridges and trees. We use haar cascade classifier Graphical User Interface (GUI) to train the images.

Vehicle Identification or detection refers the ability of computer and software system to recognize the object in an image or scene and recognize the object. The vehicle detection is done using haar cascade trained file and the input is taken as a video file which can be taken from YouTube and any CCTV surveillance camera. To increase the efficiency of the algorithm we use image processing techniques such as background subtraction and shadow removal.

Most important part is to track the vehicle. Tracking is a process of giving and ID for the detected object to know the same object in next frame. Tracking is done using dlib library. Dlibs main function is to detect multiple objects at same time. Tracking Speed estimation is done by calculating the distance travelled by the object for two consecutive frames. Speed is calculated using Euclidean distance and get the result in pixel and convert it to pixel per meter (PPM). To calculate the distance of vehicle travelled in two consecutive frames using Euclidean distance, Let $C_t(a, b)$ and $C_{t+1}(c, d)$ is centroid point of the object in frame t and $t+1$ respectively.

The distance d calculated by Euclidean distance is given below:

$$d = \sqrt{[(a - c)^2 + (b - d)^2]}$$

From the above Euclidean calculation, d_pixels are the distance travelled by the object between frames. Hence for the standard conversion i.e., from converting Pixels to meter d_pixels need to converted to d_meters ,

$$d_meters = d_pixels / ppm$$

So, to calculate the final speed of the vehicle, in Km/hr. we need to calculate the fps. This can be done by using video processing technique in preprocessing stage.

$$Speed = d_meters * fps * 3.6$$

which gives the speed of the vehicle in km/hr.

3.6 The overview of system

- I) Input as a video file and Image acquisition. First stage of video processing is the computer vision system is the image acquisition stage. After image is obtained, we can carry out number of methods to get desired output.
- II) Machine Learning: Vehicle Identification refers to trace object in a frame. At this stage we collect the data set of vehicle and non-vehicle.

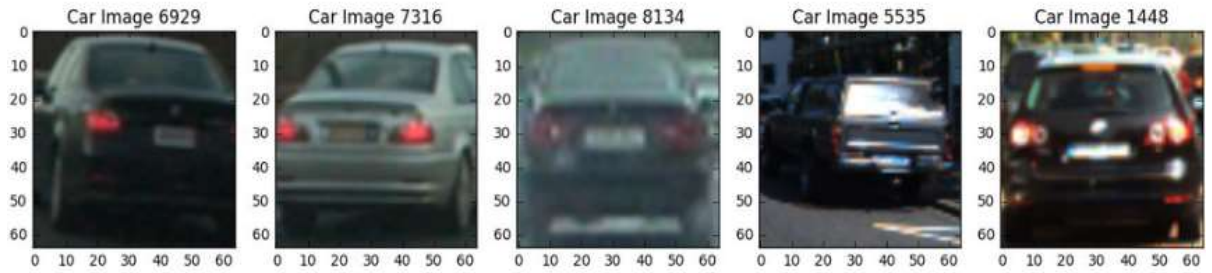


FIG. 3.1 VEHICLE IMAGE

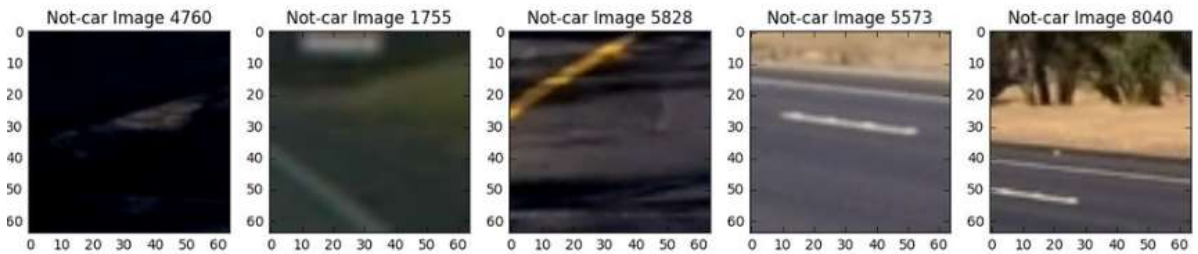


FIG. 3.2 NON- VEHICLE IMAGE

- III) Tracking: Tracking gives id to detected object. This can be done by dlib library. It is used to track the multiple objects at same time.
- IV) Speed estimation: Speed is calculate using Euclidean distance by calculating distance for two consecutive frames.

CHAPTER 4

Chapter 4

4.1 Tool Description

4.1.1 Software

A PYTHON

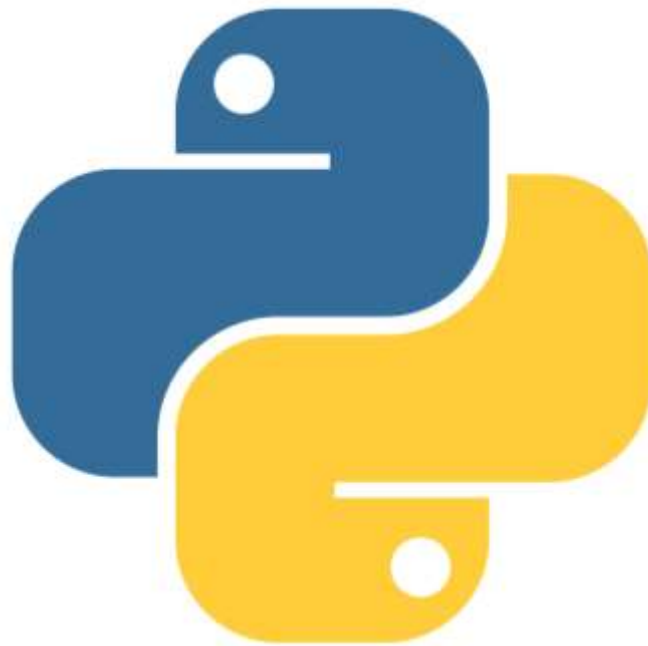


FIG. 4.1 PYTHON LOGO

Python is a computer programming language often used to build websites and software, automate tasks, and conduct data analysis. Python is a general-purpose language, meaning it can be used to create a variety of different programs and isn't specialized for any specific problems. This versatility, along with its beginner-friendliness, has made it one of the most-used programming languages today. A survey conducted by industry analyst firm Red Monk found that it was the second-most popular programming language among developers in 2021.

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built-in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

Often, programmers fall in love with Python because of the increased productivity it provides. Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the

interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace. A source level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time, and so on. The debugger is written in Python itself, testifying to Python's introspective power. On the other hand, often the quickest way to debug a program is to add a few print statements to the source: the fast edit-test-debug cycle makes this simple approach very effective.

The name Python comes from Monty Python. When Guido Van Rossum was creating Python, he was also reading the scripts from BBC's Monty Python's Flying Circus. He thought the name Python was appropriately short and slightly mysterious.

- **Features of python**

- Easier access to debuggers through a new breakpoint () built-in.
- Simple class creation using data classes.
- Customized access to module attributes.
- Improved support for type hinting.
- Higher precision timing functions.

- **What is Python used for?**

Python is commonly used for developing websites and software, task automation, data analysis, and data visualization. Since it's relatively easy to learn, Python has been adopted by many non-programmers such as accountants and scientists, for a variety of everyday tasks, like organizing finances.

"Writing programs is a very creative and rewarding activity," says University of Michigan and Coursera instructor Charles R Severance in his book *Python for Everybody*. "You can write programs for many reasons, ranging from making your living to solving a difficult data analysis problem to having fun to helping someone else solve a problem."

- **What can you do with python?**

- Data analysis and machine learning.
- Web development.
- Automation or scripting.
- Software testing and prototyping.
- Everyday tasks.

Here's a closer look at some of these common ways Python is used.

B SPYDER

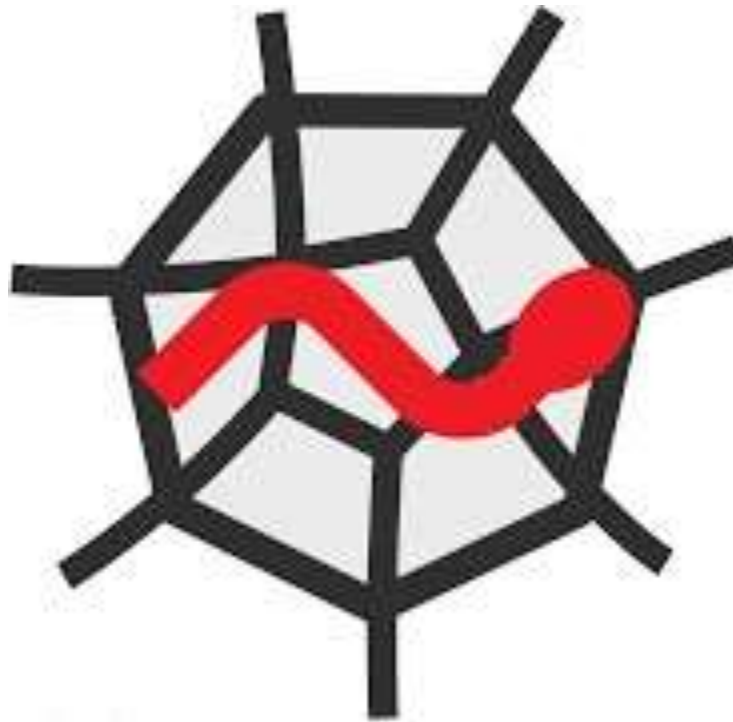


FIG. 4.2 SPYDER LOGO

Spyder is an open-source cross-platform integrated development environment (IDE) for scientific programming in the Python language. Spyder integrates with a number of prominent packages in the scientific Python stack, including NumPy, SciPy, Matplotlib, pandas, IPython, SymPy and Cython, as well as other open-source software. It is released under the MIT license.

Initially created and developed by Pierre Raybaut in 2009, since 2012 Spyder has been maintained and continuously improved by a team of scientific Python developers and the community. Spyder is extensible with first-party and third-party plugins, includes support for interactive tools for data inspection and embeds Python-specific code quality assurance and introspection instruments, such as Pyflakes, Pylint and Rope. It is available cross-platform through Anaconda, on Windows, on macOS through MacPorts, and on major Linux distributions such as Arch Linux, Debian, Fedora, Gentoo Linux, openSUSE and Ubuntu.

Spyder uses Qt for its GUI and is designed to use either of the PyQt or PySide Python bindings. QtPy, a thin abstraction layer developed by the Spyder project and later adopted by multiple other packages, provides the flexibility to use either backend

Spyder is a free and open-source scientific environment written in Python, for Python, and designed by and for scientists, engineers and data analysts. It features a unique combination of the advanced editing, analysis, debugging, and profiling functionality of a comprehensive development tool with the data exploration, interactive execution, deep inspection, and beautiful visualization capabilities of a scientific package.

Spyder is a powerful scientific environment written in Python, for Python, and designed by and for scientists, engineers and data analysts. It features a unique combination

of the advanced editing, analysis, debugging, and profiling functionality of a comprehensive development tool with the data exploration, interactive execution, deep inspection, and beautiful visualization capabilities of a scientific package.

Spyder is a cross-platform, open-source ide (IDE) for scientific Python programming. Spyder works with a variety of popular Python packages, like NumPy, Matplotlib, pandas, SymPy, and Cython, and other open-source applications. It's made available under the MIT licence. Spyder can be extended using first- and third-party plugins, and it features interactive data inspection tools as well as Python-specific code quality assurance as well as introspection tools like Pyflakes, Pylint, and Rope. Anaconda makes it cross-platform, including versions for Windows, MacOS, and major Linux distributions like Arch Linux, Debian, openSUSE, and Ubuntu. Spyder's GUI is built on Qt, and it may be used with either the PyQt or PySide Python bindings. The ability to utilise either backend is provided by QtPy, a thin abstraction layer created by the Spyder project & later adopted by a number of other programs.

- **Features include:**

- An editor with syntax highlighting, introspection, code completion.
- Support for multiple IPython consoles.
- The ability to explore and edit variables from a GUI.
- A Help pane able to retrieve and render rich text documentation on functions, classes and methods automatically or on-demand.
- A debugger linked to IPdb, for step-by-step execution.
- Static code analysis, powered by Pylint.
- A run-time Profiler, to benchmark code.
- Project support, allowing work on multiple development efforts simultaneously.
- A built-in file explorer, for interacting with the filesystem and managing projects.
- A "Find in Files" feature, allowing full regular expression search over a specified scope.
- An online help browser, allowing users to search and view Python and package documentation inside the IDE.
- A history log, recording every user command entered in each console.
- An internal console, allowing for introspection and control over Spyder's own operation.

C OpenCV

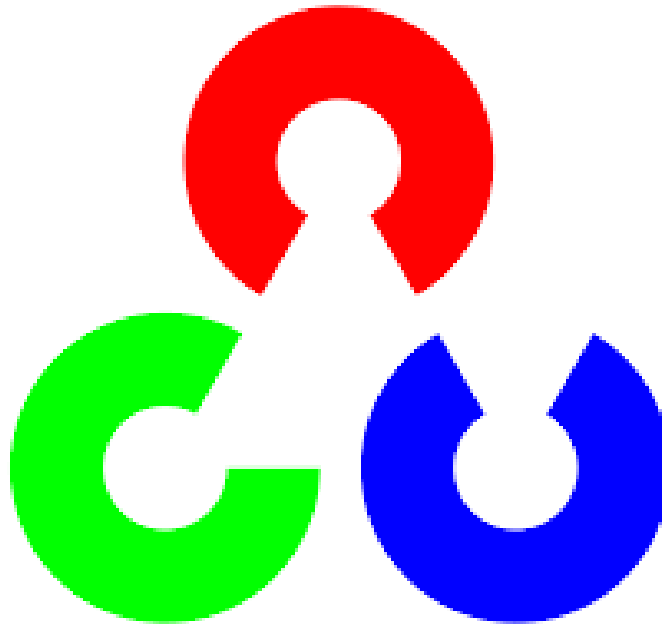


FIG. 4.3 OPENCV LOGO

OpenCV (Open-Source Computer Vision Library) is an open-source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code.

The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc. OpenCV has more than 47 thousand people of user community and estimated number of downloads exceeding 18 million. The library is used extensively in companies, research groups and by governmental bodies.

Along with well-established companies like Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, Toyota that employ the library, there are many start-ups such as Applied Minds, VideoSurf, and Zeitera, that make extensive use of OpenCV. OpenCV's deployed uses span the range from stitching street view images together, detecting intrusions in surveillance video in Israel, monitoring mine equipment in China, helping robots navigate and pick up objects at Willow Garage, detection of swimming pool drowning accidents in Europe, running interactive art in Spain and New York, checking runways for debris in Turkey, inspecting labels on products in factories around the world on to rapid face detection in Japan.

It has C++, Python, Java and MATLAB interfaces and supports Windows, Linux, Android and Mac OS. OpenCV leans mostly towards real-time vision applications and takes advantage of MMX and SSE instructions when available. A full-featured CUDA and OpenCL interfaces are being actively developed right now. There are over 500 algorithms and about 10 times as many functions that compose or support those algorithms. OpenCV is written natively in C++ and has a templated interface that works seamlessly with STL containers.

OpenCV is an open-source library that includes several hundreds of computer vision algorithms. The document describes the so-called OpenCV 2.x API, which is essentially a C++ API, as opposed to the C-based OpenCV 1.x API (C API is deprecated and not tested with "C" compiler since OpenCV 2.4 releases).

OpenCV has a modular structure, which means that the package includes several shared or static libraries. The following modules are available:

- **Core functionality (core)** - a compact module defining basic data structures, including the dense multi-dimensional array Mat and basic functions used by all other modules.
- **Image Processing (imgproc)** - an image processing module that includes linear and non-linear image filtering, geometrical image transformations (resize, affine and perspective warping, generic table-based remapping), color space conversion, histograms, and so on.
- **Video Analysis (video)** - a video analysis module that includes motion estimation, background subtraction, and object tracking algorithms.
- **Camera Calibration and 3D Reconstruction (calib3d)** - basic multiple-view geometry algorithms, single and stereo camera calibration, object pose estimation, stereo correspondence algorithms, and elements of 3D reconstruction.
- **2D Features Framework (features2d)** - salient feature detectors, descriptors, and descriptor matchers.
- **Object Detection (objdetect)** - detection of objects and instances of the predefined classes (for example, faces, eyes, mugs, people, cars, and so on).
- **High-level GUI (highgui)** - an easy-to-use interface to simple UI capabilities.
- **Video I/O (videoio)** - an easy-to-use interface to video capturing and video codecs.

Some other helper modules, such as FLANN and Google test wrappers, Python bindings, and others.

The further chapters of the document describe functionality of each module. But first, make sure to get familiar with the common API concepts used thoroughly in the library.

D DLib



FIG. 4.4 DLIB LOGO

Dlib is a general-purpose cross-platform open-source software library written in the C++ programming language. Its design is heavily influenced by ideas from design by contract and component-based software engineering. This means it is, first and foremost, a collection of independent software components, each accompanied by extensive documentation and thorough debugging modes.

Davis King has been the primary author of dlib since development began in 2002. In that time dlib has grown to include a wide variety of tools. In particular, it now contains software components for dealing with networking, threads, graphical interfaces, complex data structures, linear algebra, statistical machine learning, image processing, data mining, XML and text parsing, numerical optimization, Bayesian networks, and numerous other tasks. In recent years, much of the development has been focused on creating a broad set of statistical machine learning tools. However, dlib remains a general-purpose library and welcomes contributions of high-quality software components useful in any domain.

Core to the development philosophy of dlib is a dedication to portability and ease of use. Therefore, all code in dlib is designed to be as portable as possible and similarly to not require a user to configure or install anything. To help achieve this, all platform specific code is confined inside the API wrappers. Everything else is either layered on top of those wrappers or is written in pure ISO standard C++. Currently the library is known to work on OS X, MS Windows, Linux, Solaris, the BSDs, and HP-UX. It should work on any POSIX platform but I haven't had the opportunity to test it on any others (if you have access to other platforms and would like to help increase this list then let me know).

The rest of this page explains everything you need to know to get started using the library. It explains where to find the documentation for each object/function and how to interpret what you find there. For help compiling with dlib check out the how to

compile page. Or if you are having trouble finding where a particular object's documentation is located you may be able to find it by consulting the index.

E VISUAL STUDIO CODE



FIG. 4.5 VISUAL STUDIO CODE LOGO

Visual Studio is an **Integrated Development Environment (IDE)** developed by Microsoft to develop GUI (Graphical User Interface), console, Web applications, web apps, mobile apps, cloud, and web services, etc. With the help of this IDE, you can create managed code as well as native code. It uses the various platforms of Microsoft software development software like Windows store, Microsoft Silverlight, and Windows API, etc. It is not a language-specific IDE as you can use this to write code in C#, C++, VB (Visual Basic), Python, JavaScript, and many more languages. It provides support for 36 different programming languages. It is available for Windows as well as for macOS.

An IDE is a feature-rich program that supports many aspects of software development. The Visual Studio IDE is a creative launching pad that you can use to edit, debug, and build code, and then publish an app. Over and above the standard editor and debugger that most IDEs provide, Visual Studio includes compilers, code completion tools, graphical designers, and many more features to enhance the software development process.

Visual Studio includes a code editor supporting IntelliSense (the code completion component) as well as code refactoring. The integrated debugger works both as a source-level debugger and a machine-level debugger. Other built-in tools include a code profiler, designer for building GUI applications, web designer, class designer, and database schema designer. It accepts plug-ins that expand the functionality at almost every level—including adding support for source control systems (like Subversion and Git) and adding new toolsets like editors and visual designers for domain-specific languages or toolsets for

other aspects of the software development lifecycle (like the Azure DevOps client: Team Explorer).

Visual Studio Code is a lightweight but powerful source code editor which runs on your desktop and is available for Windows, macOS and Linux. It comes with built-in support for JavaScript, TypeScript and Node.js and has a rich ecosystem of extensions for other languages (such as C++, C#, Java, Python, PHP, Go) and runtimes (such as .NET and Unity). Begin your journey with VS Code with these introductory videos.

Out of the box, Visual Studio Code includes basic support for most common programming languages. This basic support includes syntax highlighting, bracket matching, code folding, and configurable snippets. Visual Studio Code also ships with IntelliSense for JavaScript, TypeScript, JSON, CSS, and HTML, as well as debugging support for Node.js. Support for additional languages can be provided by freely available extensions on the VS Code Marketplace.

F JUPYTER NOTEBOOK



FIG. 4.6 JUPYTER NOTEBOOK LOGO

Jupyter notebook is an open-source, interactive web application that allows users to create and share documents that contain interactive calculations, code, images, etc. *Users can combine data, code, and visualizations into a single notebook, and create interactive “stories” that they can edit and share.* Notebooks are documents which contain both computer code (such as Python) and other text elements such as paragraph, markdown, figures, links, etc. The Jupyter notebook is widely used and well documented and offers an easy-to-use interface for creating, editing, and running notebooks. The notebook runs as a

web application called the “Dashboard” or “control panel” that shows local files and allows users to open notebook documents and run snippets of code. The outputs are neatly formatted and displayed on the browser.

The other component of the notebook is the kernel. The kernel is a “computational engine” that executes the code written in the Notebook. It is similar to the back-end of the application. The IPython kernel (Jupyter was previously called IPython notebook) is used to execute Python code in the Jupyter notebook. There are kernels for other languages as well, but in this article, we will explore running Python code in the notebook.

Jupyter notebooks have seen enthusiastic adoption in the data science community, to an extent where they are increasingly replacing Microsoft Word as the default authoring environment for research. Within digital humanities literature, one can find references to Jupyter notebooks (split off from *iPython*, or interactive Python, notebooks in 2014) dating to 2015.

Jupyter Notebooks have also gained traction within digital humanities as a pedagogical tool. Multiple Programming Historian tutorials such as Text Mining in Python through the HTRC Feature Reader, and Extracting Illustrated Pages from Digital Libraries with Python, as well as other pedagogical materials for workshops, make reference to putting code in a Jupyter notebook or using Jupyter notebooks to guide learners while allowing them to freely remix and edit code. The notebook format is ideally suited for teaching, especially when students have different levels of technical proficiency and comfort with writing and editing code.

The purpose of Jupyter notebooks is to provide a more accessible interface for code used in digitally-supported research or pedagogy. Tools like Jupyter notebooks are less meaningful to learn or teach about in a vacuum, because Jupyter notebooks themselves don’t *do* anything to directly further research or pedagogy. Before you start this lesson, think about what you want to get from using Jupyter Notebooks. Do you want to organize your project workflow? Do you want to work through analysing your data, keeping track of the things you try along the way? Do you want readers of your scholarship to be able to follow the theoretical and technical sides of your argument without switching between a PDF and a folder of scripts? Do you want to teach programming workshops that are more accessible to attendees with a range of technical backgrounds? Do you want to use or adapt notebooks that other people have written? Keep your goal in mind as you work through this lesson: depending on how you imagine using Jupyter notebooks, you may be able to skip sections that are mostly applicable in another context.

G ANACONDA



FIG. 4.7 ANACONDA LOGO

Anaconda is a distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment. The distribution includes data-science packages suitable for Windows, Linux, and macOS. It is developed and maintained by Anaconda, Inc., which was founded by Peter Wang and Travis Oliphant in 2012. As an Anaconda, Inc. product, it is also known as **Anaconda Distribution** or **Anaconda Individual Edition**, while other products from the company are Anaconda Team Edition and Anaconda Enterprise Edition, both of which are not free.

Package versions in Anaconda are managed by the package management system *conda*. This package manager was spun out as a separate open-source package as it ended up being useful on its own and for things other than Python. There is also a small, bootstrap version of Anaconda called **Miniconda**, which includes only conda, Python, the packages they depend on, and a small number of other packages.

Anaconda distribution comes with over 250 packages automatically installed, and over 7,500 additional open-source packages can be installed from PyPI as well as the conda package and virtual environment manager. It also includes a GUI, **Anaconda Navigator**, as a graphical alternative to the command-line interface (CLI).

The big difference between conda and the pip package manager is in how package dependencies are managed, which is a significant challenge for Python data science and the reason conda exists.

H CASCADE TRAINER GUI



FIG. 4.8 CASCADE TRAINER GUI LOGO

Cascade Trainer GUI is a program that can be used to train, test and improve cascade classifier models. It uses a graphical interface to set the parameters and make it easy to use OpenCV tools for training and testing classifiers.

Working with a boosted cascade of weak classifiers includes two major stages: the training and the detection stage. The detection stage using either HAAR or LBP based models, is described in the **object detection tutorial**. This documentation gives an overview of the functionality needed to train your own boosted cascade of weak classifiers. The current guide will walk through all the different stages: collecting training data, preparation of the training data and executing the actual model training.

Cascading is a particular case of ensemble learning based on the concatenation of several classifiers, using all information collected from the output from a given classifier as additional information for the next classifier in the cascade. Unlike voting or stacking ensembles, which are multiexpert systems, cascading is a multistage one.

Cascading classifiers are trained with several hundred "positive" sample views of a particular object and arbitrary "negative" images of the same size. After the classifier is trained it can be applied to a region of an image and detect the object in question. To search for the object in the entire frame, the search window can be moved across the image and check every location with the classifier. This process is most commonly used in image processing for object detection and tracking, primarily facial detection and recognition.

4.1.2 Hardware

4.1.2.1 Hardware Requirements

Laptop/computer A laptop computer is a small personal computer. They are designed to be more portable than traditional desktop computers, with many of the same abilities. Laptops are able to be folded flat for transportation and have a built-in keyboard and touchpad. Most laptops are powerful enough for everyday business administrative, home, or school use. However, if a user does graphical work such as 3D rendering or movie encoding, a more advanced and powerful laptop is needed. As advanced as laptops are, the top-end ones still cannot compete with high powered desktops and workstations when processing power is needed.

- **Non-functional requirements**

Non-functional requirements are the functions offered by the system. It includes time constraints and constraints on the development process and standards.

The non-functional requirements are as follows:

- **Speed:** The system should process the given input into output within appropriate time.
- **Ease of use:** The software should be user friendly. Then the customers can use easily, so it doesn't require much training time.
- **Reliability:** The rate of failures should be less then only the system is more reliable.
- **Portability:** It should be easy to implement in any system.

- **Specific Requirements**

The specific requirements are:

- **User Interfaces:** The external users are the clients. All the clients can use this software for indexing and searching.
- **Hardware Interfaces:** The external hardware interface used for indexing and searching is personal computers of the clients. The PC's may be laptops with wireless LAN as the internet connections provided will be wireless.
- **Performance Requirements:** The PC's used must be at least Pentium 4 machines so that they can give optimum performance of the product.
- **Software Interfaces:** The Operating Systems can be any version of Windows.

CHAPTER 5

Chapter 5

5.1 RESULTS AND DISCUSSION

This is software where we train the dataset.

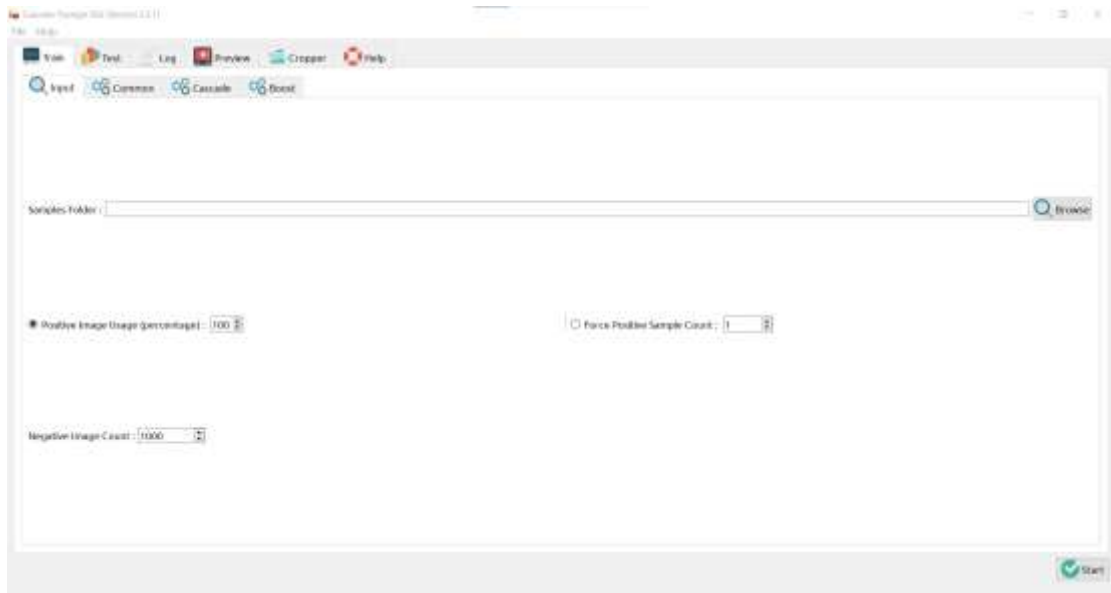


FIG. 4.1 GUI APP OF CASCADE TRAINER

Create a folder which contains sub folder named p and n. p is positive image and n is negative image. Then provide path to the folder.

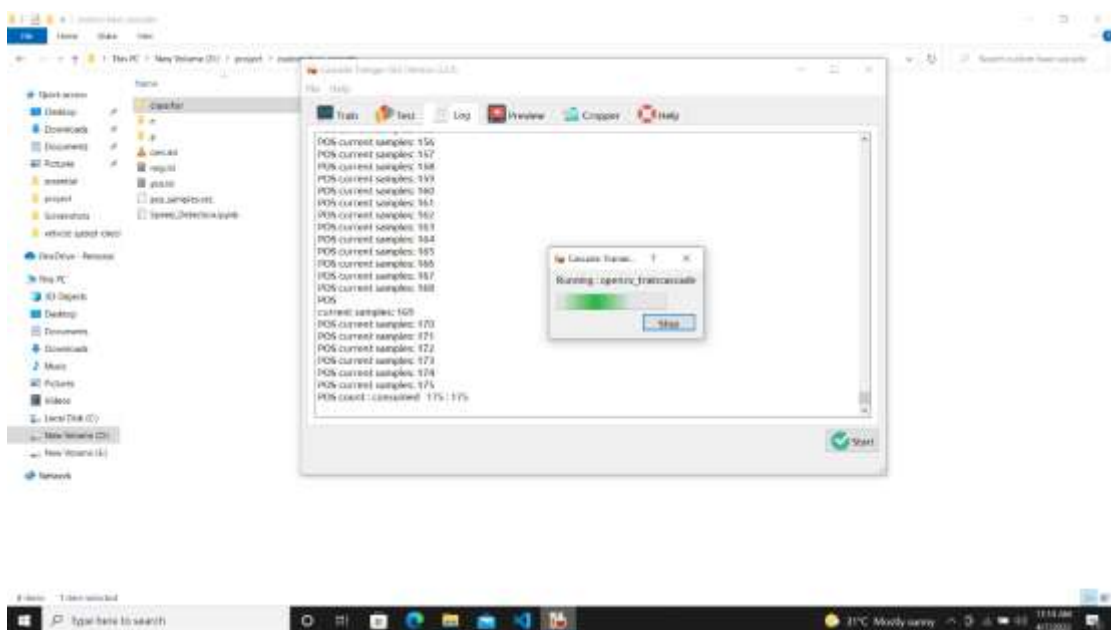


FIG. 4.2 TRAINING PHASE

When training is completed

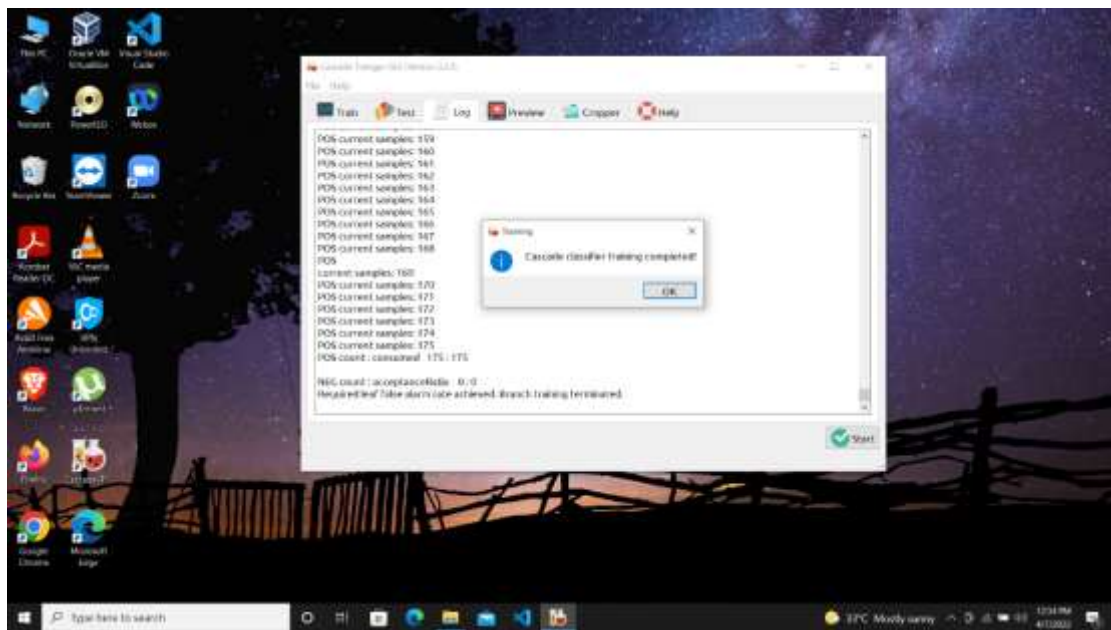


FIG. 4.3 TRAINING COMPLETED

Here figure below we have successfully detected the object.

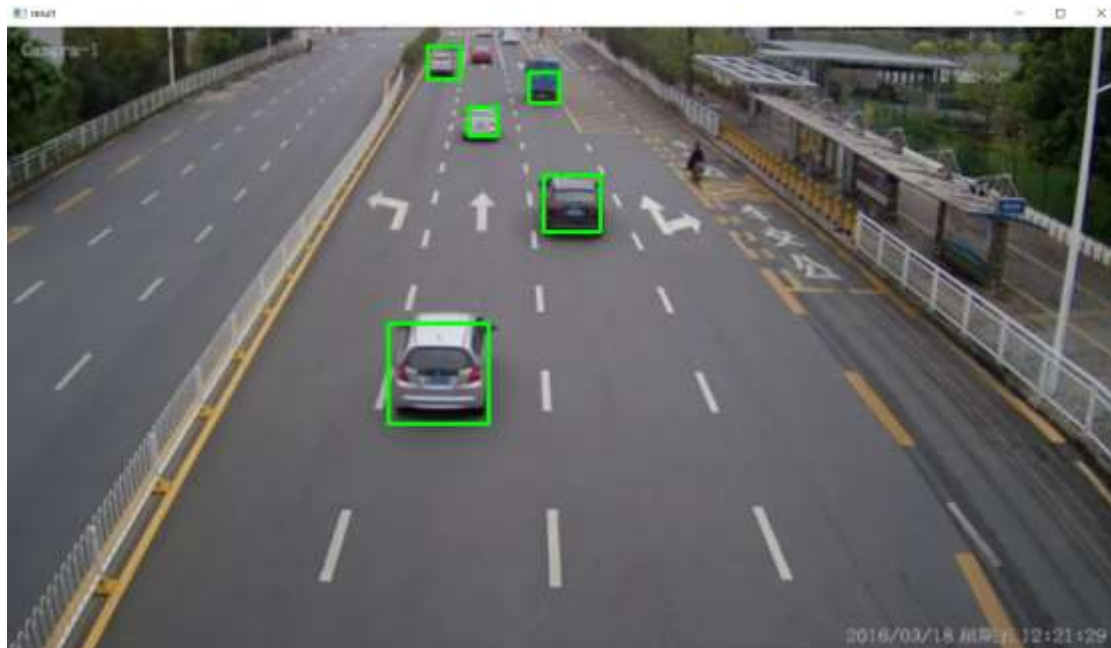


FIG. 4.4 OBJECT DETECTION

Now the most important step is to give IDs to cars so that the same vehicle can be known in the next frame.

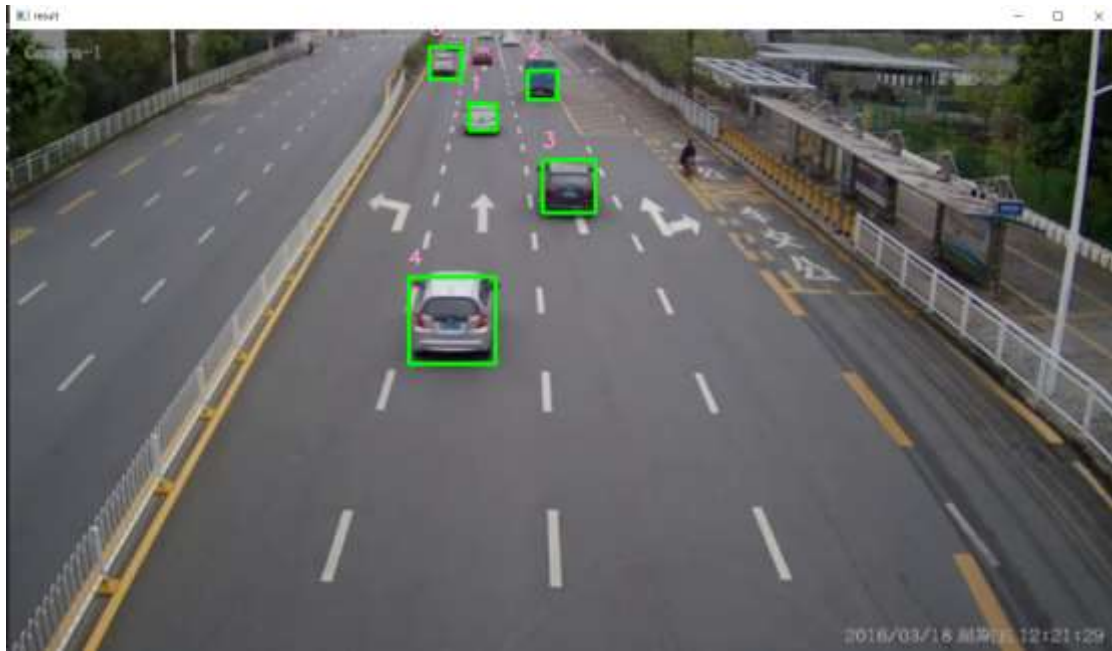


FIG. 4.5 OBJECT TRACKING

In the image above cars are associated by IDs. It will help to keep the record of the vehicle.

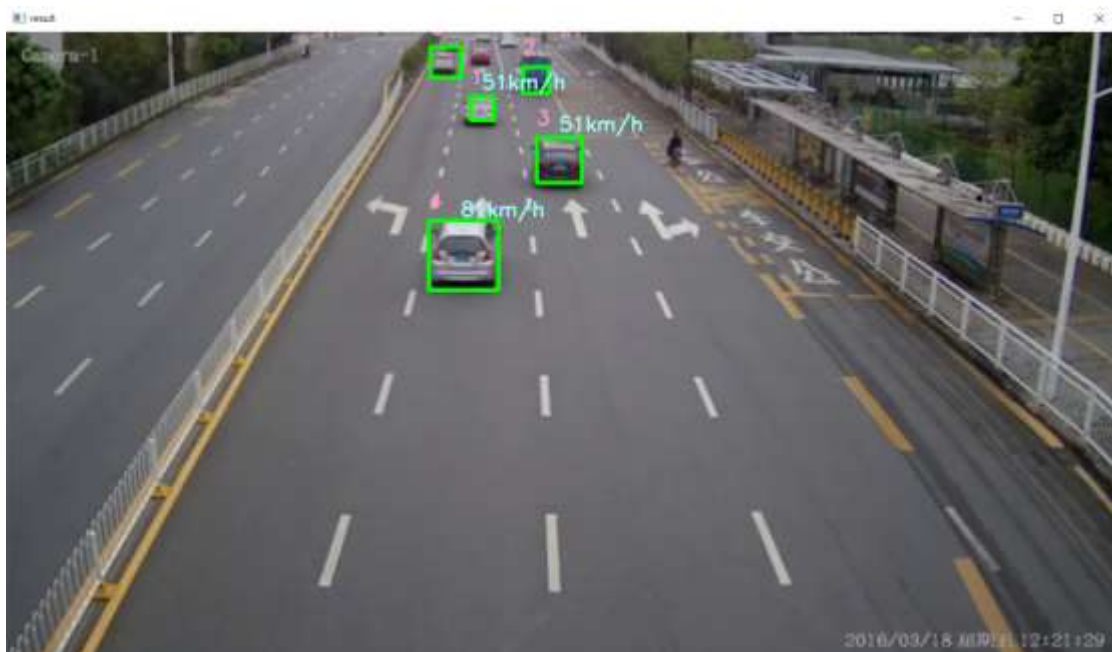


FIG. 4.6 SPEED ESTIMATION

Here the speed is calculated. If the vehicle over speeds the we instantly click the picture of vehicle for further enquiry. These save files can be useful for keeping the record.

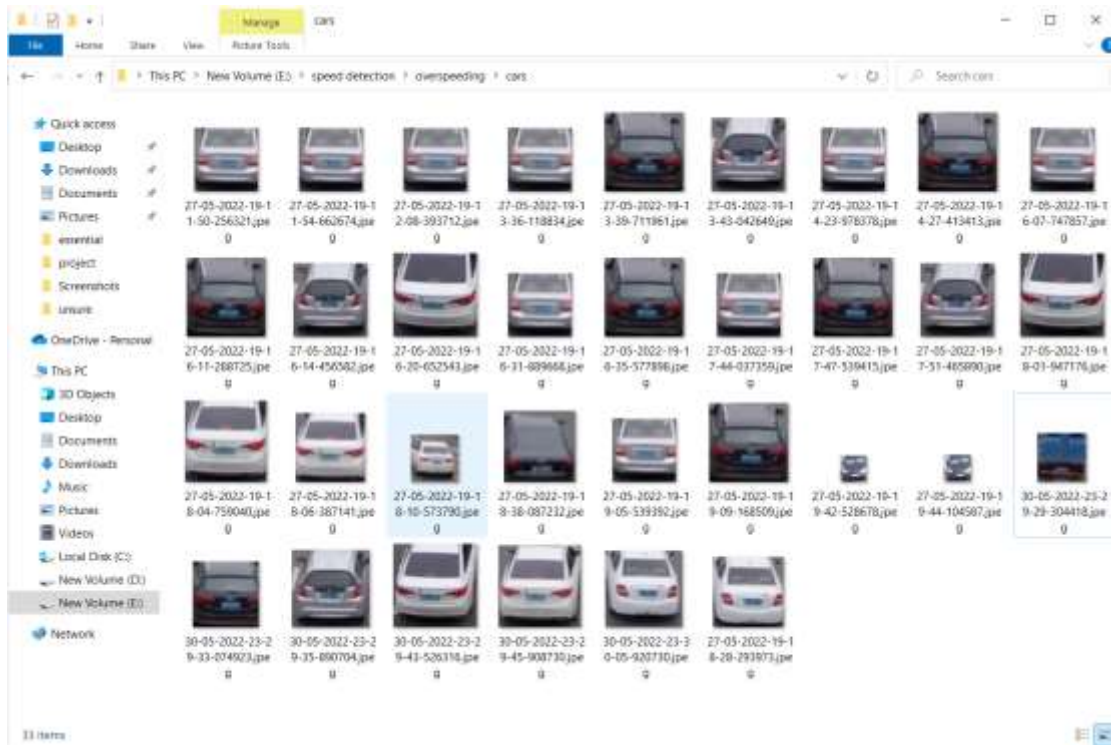


FIG. 4.7 RECORD OF OVER-SPEEDING VEHICLES

CHAPTER 6

Chapter 6

6.1 CONCLUSIONS AND FUTURE SCOPE

6.1.1 Conclusions

Road safety and reducing accidents is a very crucial issue and must be considered at utmost priority. One must hold to the rules of maintaining appropriate speed guidelines. Technological tools and tracking devices can be helpful 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 project, we build the speed detection of vehicle using machine learning algorithm and Image processing techniques like background subtraction, shadow removal. We have used Haar Cascade algorithm to detect the vehicle. Further to detect the speed of the vehicle we have used Euclidean formula. To calculate the speed, we get information about the two consecutive frames and calculate the time difference from those frames and get the distance in pixel (d_pixel) and to convert in meters we use pixel per meter (PPM) and covert in to distance in meter (d_meter) and calculate the speed of vehicle. This is the way we can detect the speed of vehicle. It can help to reduce manpower and follow Intelligent Traffic System (ITS) which is more economical and conventional method.

6.1.2 Future Scope

According to our assumption the given proposal has a very good scope in the future. The system is trying to identify the behaviour of the driver to determine if he is moving with his/her actual speed under the vision of the camera or trying to make a fool of the system. Thus, training the system in this manner for some consecutive years the system could be developed to take its own decision and record the data corresponding to different drivers or vehicles being passed regarding the actual or approximate speed with which they would be driven. The system till some time can be fed with the data being recorded treated as training data on which it would be made to carry on complete analysis to extract the features or characteristics of the driver each time being recorded or analysed with its corresponding speed and then the same data being developed would be applied in the future recorded videos which will be treated as testing data. In this way, at that time the system will require no intervention of humans. With its own intelligence it will be able to determine or prepare the complete database with the information corresponding to each driver it will make an analysis and then can send instant signal or any notification if any vehicle being captured or predicted as getting over speeded. Due to such a capability of the software the hassle to install unlimited cameras or software's at different locations in any specific area will be reduced to install a few only at required locations thus saving money and efforts.

REFERENCES

- [1] R. Reulke, A. Børner, H. Hetzheim, A. Schischmanow, and H. Venus. A sensor web for road traffic observation. In *Image and Vision Computing*, New Zealand 2002, pages 293–298, 2002.
- [2] I. Ikeda, S. Ohnaka, and M. Mizoguchi. Traffic measurement with a roadside vision system - individual tracking of overlapped vehicles. In *Proceedings of the 13th International Conference on Pattern Recognition*, pages 859–864, 1996
- [3] M. Betke, E. Haritaoglu, and L. S. Davis. Realtime multiple vehicle detection and tracking from a moving vehicle. *Machine Vision and Applications*, 12(2):69–83, 2000
- [4] M. Arai, A. Otuki, K. Nakamiira, H. Inoue, Y. Satoh, T. Iitamura, and Y. Kohayashi. Intelligent. monitoring system based on neural network theory and image processing. *Proc. ITS'95 Yobohama*, vol. 1:170-173, 1995
- [5] Adnan, Muhammad Akram, Norliana Sulaiman, Nor Izzah Zainuddin, and Tuan Badrul Hisyam Tuan Besar. "Vehicle speed measurement technique using various speed detection instrumentation." In *Business Engineering and Industrial Applications Colloquium (BEIAC)*, 2013 IEEE, pp. 668-672. IEEE, 2013
- [6] Chen, Peijiang. "Moving object detection based on background extraction." In *Computer Network and Multimedia Technology*, 2009. CNMT 2009. International Symposium on, pp. 1-4. IEEE, 2009.
- [7] Dailey, Daniel J., Fritz W. Cathey, and Suree Pumrin. "An algorithm to estimate mean traffic speed using uncalibrated cameras." *IEEE Transactions on Intelligent Transportation Systems* 1, no. 2 (2000): 98-107.
- [8] Madasu, Vamsi Krishna, and Madasu Hanmandlu. "Estimation of vehicle speed by motion tracking on image sequences." In *Intelligent Vehicles Symposium (IV)*, 2010 IEEE, pp. 185-190. IEEE, 2010.
- [9] B.Suresh, K. Triveni Y. V. Lakshmi, P. Saritha, K. Sriharsha, D. Srinivas Reddy, Determination of Moving Vehicle Speed using Image Processing, *International Journal of Engineering Research & Technology (IJERT)* ISSN: 2278-0181 Published by, www.ijert.org NCACSPV – 2016 Conference Proceedings.
- [10] Pornpanomchai, C., & Kongkittisan, K. (2009). Vehicle speed detection system. 2009 IEEE International Conference on signal and Image Processing Applications. doi:10.1109/icsipa.2009.5478629.
- [11] M. Arai, A. Otuki, K. Nakamiira, H. Inoue, Y. Satoh, T. Iitamura, and Y. Kohayashi. Intelligent. monitoring system based on neural network theory and image processing. *Proc. ITS'95 Yobohama*, vol. 1:170-173, 1995.
- [12] M. Piccardi, "Background subtraction techniques: a review", *International Conference on Systems, Man and Cybernetics*, 2004 IEEE, Vol. 4, pp. 3099-3104, 10-13 October 2004.
- [13] Amit Ghosh, Md. Shahinuzzaman Sabuj, Hamudi Hasan Sonet, Swakkhar Shatabda and Md.Farid, “ An adaptive video-based vehicle detection, classification, counting and speed measurement for real time traffic data

collection”.

- [14] O. Ibrahim, H. ElGendy and A. M. ElShafee, "Towards speed detection camera system for a radar alternative", 11th International Conference on ITS Telecommunications, 2011 IEEE, ITST 2011, pp. 627-632, St. Petersburg 23-25 August 2011.
- [15] Asaad AMA, Syed IA (2009). "Object identification in video images using morphological background estimation scheme" Chapter, 22: 279-288.
- [16] R. T. Collins, A. J. Lipton, T. Kanade, H. Fujiyoshi, D. Duggins, Y. Tsin, D. Tolliver, N. Enomoto, O. Hasegawa, P. Burt and L. Wixson, "The Robotics Institute", Carnegie Mellon University, Pittsburgh PA The Sarnoff Corporation, Princeton, NJ.
- [17] E. Atkociunas, R. Blake, A. Juozapavicius, M. Kazimianec, Image Processing in Road Traffic Analysis, Nonlinear Analysis: Modelling and Control, 2005, Vol. 10, No. 4, 315-33.
- [18] N. Prabhakar, V. Vaithyanathan, A. P. Sharma, A. Singh and P. Singhal, "Object Tracking Using Frame Differencing and Template Matching", Research Journal of Applied Sciences, Engineering and Technology, Vol. 4(24) pp. 5497-5501, December 2012.
- [19] Jin-xiang Wang, Research of vehicle speed detection algorithm in video surveillance, IIP Lab. Department of Computer Science and Technology, Yanbian University, Yanji, Jilin, China.
- [20] Wen L, Fumio Y (2009). "Speed detection of moving vehicles from one scene of Quick Bird images", Urban Remote Sensing Joint Event, pp. 1-6.
- [21] F. Bounini, D. Gingras, V. Lapointe, H. Pollart, "Autonomous vehicle and Real time road lanes detection and tracking", Vehicle power and propulsion conference, 2015 IEEE, VPPC 2015. Pp. 1-6, Montreal, 19- 22 October 2015.
- [22] R. C. Gonzalez, R. E. Woods, "Digital Image Processing" Third Edition, Prentice Hall, 2007.
- [23] T. Kumar, R. Sachan and D. S. Kushwaha, Smart City Traffic Management and Surveillance System for Indian Scenario, Proceedings of International Conference on Recent Advances in Mathematics, Statistics and Computer Science (ICRAMSCS), (2015)

APPENDIX-I

8.1 SOURCE CODE

8.1.1 Vech.xml

```
1    <?xml version="1.0"?>
2    <opencv_storage>
3    <myhaar type_id="opencv-haar-classifier">
4    <size>
5        24 24</size>
6    <stages>
7        <_>
8            <trees>
9                <_>
10               <_>
11                   <feature>
12                       <rects>
13                           <_>
14                               5 20 14 4 -1.</_>
15                           <_>
16                               5 22 14 2 2.</_></rects>
17                       <tilted>0</tilted></feature>
18                       <threshold>-1.8861599965021014e-003</threshold>
19                       <left_val>-0.7102618217468262</left_val>
20                       <right_val>0.8759310841560364</right_val></_></_>
21                   <_>
22                       <_>
23                           <feature>
24                               <rects>
25                                   <_>
26                                       6 14 12 4 -1.</_>
27                                   <_>
28                                       10 14 4 4 3.</_></rects>
29                           <tilted>0</tilted></feature>
30                           <threshold>-6.8823847686871886e-004</threshold>
31                           <left_val>0.7683624029159546</left_val>
32                           <right_val>-0.8232979774475098</right_val></_></_></trees>
33                   <stage_threshold>0.0526331104338169</stage_threshold>
34                   <parent>-1</parent>
35                   <next>-1</next></_>
36               <_>
37                   <trees>
38                       <_>
39                           <_>
40                               <feature>
41                                   <rects>
42                                       <_>
43                                           12 10 6 8 -1.</_>
44                               <_>
```

```

45         10 12 6 4 2.</_></rects>
46     <tilted>1</tilted></feature>
47     <threshold>-0.0143376495689154</threshold>
48     <left_val>0.8701300024986267</left_val>
49     <right_val>-0.6504855155944824</right_val></_></_>
50 <_>
51 <_>
52     <feature>
53     <rects>
54     <_>
55         23 1 1 4 -1.</_>
56     <_>
57         23 3 1 2 2.</_></rects>
58     <tilted>0</tilted></feature>
59     <threshold>1.4198479475453496e-003</threshold>
60     <left_val>0.3970716893672943</left_val>
61     <right_val>-0.7556182742118835</right_val></_></_>
62 <_>
63 <_>
64     <feature>
65     <rects>
66     <_>
67         5 13 12 4 -1.</_>
68     <_>
69         8 13 6 4 2.</_></rects>
70     <tilted>0</tilted></feature>
71     <threshold>-4.3507227674126625e-003</threshold>
72     <left_val>0.9354869127273560</left_val>
73     <right_val>-0.6028910279273987</right_val></_></_>
74 <_>
75 <_>
76     <feature>
77     <rects>
78     <_>
79         16 1 8 6 -1.</_>
80     <_>
81         16 1 4 6 2.</_></rects>
82     <tilted>0</tilted></feature>
83     <threshold>0.0154949799180031</threshold>
84     <left_val>-0.1221532970666885</left_val>
85     <right_val>0.9999998211860657</right_val></_></_>
86 <_>
87 <_>
88     <feature>
89     <rects>
90     <_>
91         0 1 8 6 -1.</_>
92     <_>

```

```

93         4 1 4 6 2.</_></rects>
94     <tilted>0</tilted></feature>
95     <threshold>-3.5563390702009201e-003</threshold>
96     <left_val>-0.9594147801399231</left_val>
97     <right_val>0.6721295714378357</right_val></_></_></trees>
98 <stage_threshold>-0.3063285052776337</stage_threshold>
99 <parent>0</parent>
100 <next>-1</next></_>
101 <_>
102 <trees>
103     <_>
104         <_>
105             <feature>
106                 <rects>
107                     <_>
108                         1 10 22 2 -1.</_>
109                     <_>
110                         12 10 11 2 2.</_></rects>
111                 <tilted>0</tilted></feature>
112                 <threshold>-2.6650010840967298e-004</threshold>
113                 <left_val>-0.7102615833282471</left_val>
114                 <right_val>0.8759310841560364</right_val></_></_>
115             <_>
116                 <_>
117                     <feature>
118                         <rects>
119                             <_>
120                                 10 6 4 18 -1.</_>
121                             <_>
122                                 12 6 2 9 2.</_>
123                             <_>
124                                 10 15 2 9 2.</_></rects>
125                         <tilted>0</tilted></feature>
126                         <threshold>1.3746969634667039e-003</threshold>
127                         <left_val>-0.5181714296340942</left_val>
128                         <right_val>1.0000020265579224</right_val></_></_>
129                     <_>
130                         <_>
131                             <feature>
132                                 <rects>
133                                     <_>
134                                         9 21 4 2 -1.</_>
135                                     <_>
136                                         10 21 2 2 2.</_></rects>
137                             <tilted>0</tilted></feature>
138                             <threshold>2.6440850342623889e-004</threshold>
139                             <left_val>0.6216245293617249</left_val>
140                             <right_val>-0.9268335103988648</right_val></_></_></trees>

```



```

141     <stage_threshold>-0.6068084836006165</stage_threshold>
142     <parent>1</parent>
143     <next>-1</next></_>
144 <_>
145     <trees>
146     <_>
147         <_>
148             <feature>
149                 <rects>
150                     <_>
151                         8 10 6 12 -1.</_>
152                     <_>
153                         8 14 6 4 3.</_></rects>
154                 <tilted>0</tilted></feature>
155             <threshold>-0.0456857085227966</threshold>
156             <left_val>1.</left_val>
157             <right_val>-0.6129032969474793</right_val></_></_>
158         <_>
159         <_>
160             <feature>
161                 <rects>
162                     <_>
163                         18 0 5 24 -1.</_>
164                     <_>
165                         18 12 5 12 2.</_></rects>
166                 <tilted>0</tilted></feature>
167             <threshold>-0.1984090954065323</threshold>
168             <left_val>-1.</left_val>
169             <right_val>0.2632923126220703</right_val></_></_>
170         <_>
171         <_>
172             <feature>
173                 <rects>
174                     <_>
175                         1 0 5 24 -1.</_>
176                     <_>
177                         1 12 5 12 2.</_></rects>
178                 <tilted>0</tilted></feature>
179             <threshold>-0.0250024907290936</threshold>
180             <left_val>0.6768069267272949</left_val>
181             <right_val>-0.8706650733947754</right_val></_></_>
182         <_>
183         <_>
184             <feature>
185                 <rects>
186                     <_>
187                         2 11 20 1 -1.</_>
188                     <_>

```

```

189         2 11 10 1 2.</_></rects>
190     <tilted>0</tilted></feature>
191     <threshold>-2.2996349725872278e-003</threshold>
192     <left_val>0.6934041976928711</left_val>
193     <right_val>-0.7547464966773987</right_val></_></_></trees>
194 <stage_threshold>-0.5268719196319580</stage_threshold>
195 <parent>2</parent>
196 <next>-1</next></_>
197 <_>
198 <trees>
199     <_>
200     <_>
201     <feature>
202     <rects>
203     <_>
204         9 11 4 12 -1.</_>
205     <_>
206         9 15 4 4 3.</_></rects>
207     <tilted>0</tilted></feature>
208     <threshold>-0.0301342792809010</threshold>
209     <left_val>1.</left_val>
210     <right_val>-0.6129032969474793</right_val></_></_>
211 <_>
212 <_>
213     <feature>
214     <rects>
215     <_>
216         21 2 2 14 -1.</_>
217     <_>
218         21 2 1 14 2.</_></rects>
219     <tilted>0</tilted></feature>
220     <threshold>-3.1865958590060472e-005</threshold>
221     <left_val>0.4441277980804443</left_val>
222     <right_val>-0.0825580805540085</right_val></_></_>
223 <_>
224 <_>
225     <feature>
226     <rects>
227     <_>
228         1 2 2 14 -1.</_>
229     <_>
230         2 2 1 14 2.</_></rects>
231     <tilted>0</tilted></feature>
232     <threshold>-1.8942370661534369e-004</threshold>
233     <left_val>-0.7303994297981262</left_val>
234     <right_val>0.7530428767204285</right_val></_></_></trees>
235 <stage_threshold>-0.8991749286651611</stage_threshold>
236 <parent>3</parent>

```

```

237     <next>-1</next></_>
238 <_>
239 <trees>
240     <_>
241     <_>
242     <feature>
243     <rects>
244     <_>
245     7 11 12 1 -1.</_>
246     <_>
247     10 14 6 1 2.</_></rects>
248     <tilted>1</tilted></feature>
249     <threshold>-0.0155694000422955</threshold>
250     <left_val>1.</left_val>
251     <right_val>-0.5625001788139343</right_val></_></_>
252 <_>
253 <_>
254 <feature>
255 <rects>
256 <_>
257 11 18 10 2 -1.</_>
258 <_>
259 16 18 5 1 2.</_>
260 <_>
261 11 19 5 1 2.</_></rects>
262 <tilted>0</tilted></feature>
263 <threshold>1.8961379828397185e-004</threshold>
264 <left_val>-0.3225373029708862</left_val>
265 <right_val>0.6139081716537476</right_val></_></_>
266 <_>
267 <_>
268 <feature>
269 <rects>
270 <_>
271 3 18 10 2 -1.</_>
272 <_>
273 3 18 5 1 2.</_>
274 <_>
275 8 19 5 1 2.</_></rects>
276 <tilted>0</tilted></feature>
277 <threshold>-3.7600338691845536e-004</threshold>
278 <left_val>-0.7823668122291565</left_val>
279 <right_val>0.7111589908599854</right_val></_></_></trees>
280 <stage_threshold>-0.1738784015178680</stage_threshold>
281 <parent>4</parent>
282 <next>-1</next></_>
283 <_>
284 <trees>

```

```

285     <_>
286     <_>
287     <feature>
288     <rects>
289     <_>
290         1 12 20 9 -1.</_>
291     <_>
292         6 12 10 9 2.</_></rects>
293     <tilted>0</tilted></feature>
294     <threshold>-0.0230577308684587</threshold>
295     <left_val>0.7613366246223450</left_val>
296     <right_val>-0.6632016897201538</right_val></_></_>
297 <_>
298 <_>
299     <feature>
300     <rects>
301     <_>
302         20 6 2 9 -1.</_>
303     <_>
304         20 6 1 9 2.</_></rects>
305     <tilted>0</tilted></feature>
306     <threshold>9.1012020129710436e-004</threshold>
307     <left_val>-0.1542842984199524</left_val>
308     <right_val>0.6441141963005066</right_val></_></_>
309 <_>
310 <_>
311     <feature>
312     <rects>
313     <_>
314         2 6 2 9 -1.</_>
315     <_>
316         3 6 1 9 2.</_></rects>
317     <tilted>0</tilted></feature>
318     <threshold>-1.2507000064942986e-004</threshold>
319     <left_val>-0.7068138122558594</left_val>
320     <right_val>0.8441488146781921</right_val></_></_></trees>
321 <stage_threshold>-0.7259013056755066</stage_threshold>
322 <parent>5</parent>
323 <next>-1</next></_>
324 <_>
325 <trees>
326 <_>
327 <_>
328     <feature>
329     <rects>
330     <_>
331         0 14 2 10 -1.</_>
332     <_>

```

```

333         0 19 2 5 2.</_></rects>
334     <tilted>0</tilted></feature>
335     <threshold>-7.1406401693820953e-003</threshold>
336     <left_val>-0.7621145844459534</left_val>
337     <right_val>0.7757853865623474</right_val></_></_>
338 <_>
339 <_>
340     <feature>
341     <rects>
342     <_>
343         3 7 18 10 -1.</_>
344     <_>
345         12 7 9 5 2.</_>
346     <_>
347         3 12 9 5 2.</_></rects>
348     <tilted>0</tilted></feature>
349     <threshold>0.0320104286074638</threshold>
350     <left_val>0.6480373740196228</left_val>
351     <right_val>-0.8101025223731995</right_val></_></_></trees>
352 <stage_threshold>-0.1140772029757500</stage_threshold>
353 <parent>6</parent>
354 <next>-1</next></_>
355 <_>
356 <trees>
357 <_>
358 <_>
359     <feature>
360     <rects>
361     <_>
362         0 5 5 12 -1.</_>
363     <_>
364         0 9 5 4 3.</_></rects>
365     <tilted>0</tilted></feature>
366     <threshold>-2.6784769725054502e-003</threshold>
367     <left_val>-0.6632016897201538</left_val>
368     <right_val>0.7613372206687927</right_val></_></_>
369 <_>
370 <_>
371     <feature>
372     <rects>
373     <_>
374         2 6 21 18 -1.</_>
375     <_>
376         9 12 7 6 9.</_></rects>
377     <tilted>0</tilted></feature>
378     <threshold>-0.3079409003257752</threshold>
379     <left_val>0.8817912936210632</left_val>
380     <right_val>-0.4137161970138550</right_val></_></_>

```

```

381     <_>
382     <_>
383     <feature>
384     <rects>
385     <_>
386     7 9 2 2 -1.</_>
387     <_>
388     7 9 1 1 2.</_>
389     <_>
390     8 10 1 1 2.</_></rects>
391     <tilted>0</tilted></feature>
392     <threshold>1.1662089673336595e-004</threshold>
393     <left_val>0.6626076102256775</left_val>
394     <right_val>-0.7856904864311218</right_val></_></_></trees>
395     <stage_threshold>-0.5671008825302124</stage_threshold>
396     <parent>7</parent>
397     <next>-1</next></_>
398 <_>
399 <trees>
400 <_>
401 <_>
402 <feature>
403 <rects>
404 <_>
405 2 1 2 1 -1.</_>
406 <_>
407 2 1 1 1 2.</_></rects>
408 <tilted>1</tilted></feature>
409 <threshold>-1.4016850036568940e-004</threshold>
410 <left_val>-0.7418035268783569</left_val>
411 <right_val>0.8786414861679077</right_val></_></_>
412 <_>
413 <_>
414 <feature>
415 <rects>
416 <_>
417 17 8 7 8 -1.</_>
418 <_>
419 17 8 7 4 2.</_></rects>
420 <tilted>1</tilted></feature>
421 <threshold>0.1201006025075913</threshold>
422 <left_val>-0.4733473956584930</left_val>
423 <right_val>1.0000020265579224</right_val></_></_>
424 <_>
425 <_>
426 <feature>
427 <rects>
428 <_>

```

```

429         0 1 1 4 -1.</_>
430     <_>
431         0 3 1 2 2.</_></rects>
432     <tilted>0</tilted></feature>
433     <threshold>3.6260599154047668e-004</threshold>
434     <left_val>0.6596605777740479</left_val>
435     <right_val>-0.8896765112876892</right_val></_></_></trees>
436 <stage_threshold>-0.6314778923988342</stage_threshold>
437 <parent>8</parent>
438 <next>-1</next></_>
439 <_>
440 <trees>
441     <_>
442     <_>
443     <feature>
444     <rects>
445     <_>
446         0 4 4 20 -1.</_>
447     <_>
448         0 4 2 10 2.</_>
449     <_>
450         2 14 2 10 2.</_></rects>
451     <tilted>0</tilted></feature>
452     <threshold>1.7956939991563559e-003</threshold>
453     <left_val>0.6987950801849365</left_val>
454     <right_val>-0.8656712770462036</right_val></_></_>
455 <_>
456 <_>
457 <feature>
458 <rects>
459 <_>
460         15 21 3 3 -1.</_>
461 <_>
462         16 21 1 3 3.</_></rects>
463     <tilted>0</tilted></feature>
464     <threshold>5.4609519429504871e-004</threshold>
465     <left_val>0.1357603073120117</left_val>
466     <right_val>-0.7273567914962769</right_val></_></_>
467 <_>
468 <_>
469 <feature>
470 <rects>
471 <_>
472         6 21 3 3 -1.</_>
473 <_>
474         7 21 1 3 3.</_></rects>
475     <tilted>0</tilted></feature>
476     <threshold>-3.3102690940722823e-004</threshold>

```

```

477         <left_val>-0.9668954014778137</left_val>
478         <right_val>0.6947696208953857</right_val></_></_></trees>
479     <stage_threshold>-0.1323399990797043</stage_threshold>
480     <parent>9</parent>
481     <next>-1</next></_>
482 <_>
483 <trees>
484 <_>
485 <_>
486     <feature>
487         <rects>
488             <_>
489                 8 5 8 11 -1.</_>
490             <_>
491                 10 5 4 11 2.</_></rects>
492         <tilted>0</tilted></feature>
493     <threshold>-2.7545159682631493e-003</threshold>
494     <left_val>1.</left_val>
495     <right_val>-0.6129029989242554</right_val></_></_>
496 <_>
497 <_>
498     <feature>
499         <rects>
500             <_>
501                 18 0 3 20 -1.</_>
502             <_>
503                 18 10 3 10 2.</_></rects>
504         <tilted>0</tilted></feature>
505     <threshold>0.0941873565316200</threshold>
506     <left_val>0.2309058010578156</left_val>
507     <right_val>-0.9999986886978149</right_val></_></_>
508 <_>
509 <_>
510     <feature>
511         <rects>
512             <_>
513                 11 13 2 4 -1.</_>
514             <_>
515                 12 13 1 4 2.</_></rects>
516         <tilted>0</tilted></feature>
517     <threshold>-2.9519940653699450e-005</threshold>
518     <left_val>-0.7264736890792847</left_val>
519     <right_val>0.8168529868125916</right_val></_></_></trees>
520 <stage_threshold>0.4348557889461517</stage_threshold>
521 <parent>10</parent>
522 <next>-1</next></_>
523 <_>
524 <trees>

```



```

525     <_>
526     <_>
527     <feature>
528     <rects>
529     <_>
530         5 8 10 2 -1.</_>
531     <_>
532         5 8 5 2 2.</_></rects>
533     <tilted>1</tilted></feature>
534     <threshold>0.0343970097601414</threshold>
535     <left_val>-0.6129032969474793</left_val>
536     <right_val>1.0000009536743164</right_val></_></_>
537 <_>
538 <_>
539 <feature>
540 <rects>
541 <_>
542     13 22 7 2 -1.</_>
543 <_>
544     13 23 7 1 2.</_></rects>
545 <tilted>0</tilted></feature>
546 <threshold>-2.2730599157512188e-003</threshold>
547 <left_val>-0.7018197178840637</left_val>
548 <right_val>0.3487352132797241</right_val></_></_>
549 <_>
550 <_>
551 <feature>
552 <rects>
553 <_>
554     1 0 18 4 -1.</_>
555 <_>
556     10 0 9 4 2.</_></rects>
557 <tilted>0</tilted></feature>
558 <threshold>0.0158242993056774</threshold>
559 <left_val>0.7520508766174316</left_val>
560 <right_val>-0.8825100064277649</right_val></_></_></trees>
561 <stage_threshold>-0.5626720786094666</stage_threshold>
562 <parent>11</parent>
563 <next>-1</next></_>
564 <_>
565 <trees>
566 <_>
567 <_>
568 <feature>
569 <rects>
570 <_>
571     0 11 24 1 -1.</_>
572 <_>

```

```

573         12 11 12 1 2.</_></rects>
574     <tilted>0</tilted></feature>
575     <threshold>-1.3968610437586904e-003</threshold>
576     <left_val>-0.7721518278121948</left_val>
577     <right_val>0.6039606928825378</right_val></_></_>
578 <_>
579 <_>
580     <feature>
581     <rects>
582     <_>
583         9 22 6 2 -1.</_>
584     <_>
585         9 23 6 1 2.</_></rects>
586     <tilted>0</tilted></feature>
587     <threshold>-1.6340760339517146e-005</threshold>
588     <left_val>-0.6897482872009277</left_val>
589     <right_val>0.8366590142250061</right_val></_></_></trees>
590 <stage_threshold>-0.0857876464724541</stage_threshold>
591 <parent>12</parent>
592 <next>-1</next></_>
593 <_>
594 <trees>
595 <_>
596 <_>
597     <feature>
598     <rects>
599     <_>
600         6 12 11 9 -1.</_>
601     <_>
602         6 15 11 3 3.</_></rects>
603     <tilted>0</tilted></feature>
604     <threshold>-0.0324200503528118</threshold>
605     <left_val>1.</left_val>
606     <right_val>-0.5384616255760193</right_val></_></_>
607 <_>
608 <_>
609     <feature>
610     <rects>
611     <_>
612         18 1 6 9 -1.</_>
613     <_>
614         18 1 3 9 2.</_></rects>
615     <tilted>0</tilted></feature>
616     <threshold>-0.0151985604315996</threshold>
617     <left_val>1.</left_val>
618     <right_val>-0.0111628798767924</right_val></_></_>
619 <_>
620 <_>

```

```

621     <feature>
622     <rects>
623     <_>
624     9 3 15 3 -1.</_>
625     <_>
626     8 4 15 1 3.</_></rects>
627     <tilted>1</tilted></feature>
628     <threshold>2.9830720741301775e-003</threshold>
629     <left_val>-0.5220050215721130</left_val>
630     <right_val>0.8720135092735291</right_val></_></_>
631 <_>
632 <_>
633 <feature>
634 <rects>
635 <_>
636 9 0 12 6 -1.</_>
637 <_>
638 9 0 6 6 2.</_></rects>
639 <tilted>0</tilted></feature>
640 <threshold>0.0109072700142860</threshold>
641 <left_val>-0.1559285074472427</left_val>
642 <right_val>0.5554535984992981</right_val></_></_>
643 <_>
644 <_>
645 <feature>
646 <rects>
647 <_>
648 10 19 3 2 -1.</_>
649 <_>
650 11 20 1 2 3.</_></rects>
651 <tilted>1</tilted></feature>
652 <threshold>3.1258648959919810e-005</threshold>
653 <left_val>-0.5868247747421265</left_val>
654 <right_val>0.9999995827674866</right_val></_></_></trees>
655 <stage_threshold>-0.4203642010688782</stage_threshold>
656 <parent>13</parent>
657 <next>-1</next></_></stages></myhaar>
658 </opencv_storage>

```

This vech.xml training file contain sample of the original file. The original code is lot bigger and is not logically possible to add those complete line.

8.1.2 Main.py

```
import cv2
import dlib
import time
import math
import os
from datetime import datetime

carCascade = cv2.CascadeClassifier('veh.xml')
video = cv2.VideoCapture('cars.mp4')

WIDTH = 1280
HEIGHT = 720
speedLimit = 40 #SPEEDLIMIT

if not os.path.exists('overspeeding/cars/'):
    os.makedirs('overspeeding/cars/')

print('Speed Limit Set at 40 Kmph')

def saveCar(speed,image):
    now = datetime.today().now()
    nameCurTime = now.strftime("%d-%m-%Y-%H-%M-%S-%f")

    link = 'overspeeding/cars/'+nameCurTime+'.jpeg'
    cv2.imwrite(link,image)

def estimateSpeed(location1, location2):
    d_pixels = math.sqrt(math.pow(location2[0] - location1[0], 2) +
math.pow(location2[1] - location1[1], 2))
    # ppm = location2[2] / carWidht
    ppm = 8.8
```

```

d_meters = d_pixels / ppm
fps = 18
speed = d_meters * fps * 3.6
return speed

def trackMultipleObjects():
    rectangleColor = (0, 255, 0)
    frameCounter = 0
    currentCarID = 0
    # fps = 0

    carTracker = {}
    # carNumbers = {}
    carLocation1 = {}
    carLocation2 = {}
    speed = [None] * 1000

    outputVideo = cv2.VideoWriter('outNew.avi',
cv2.VideoWriter_fourcc('M','J','P','G'), 10, (WIDTH, HEIGHT))

    out = cv2.VideoWriter('outNew.avi', cv2.VideoWriter_fourcc('M','J','P','G'),
10, (WIDTH, HEIGHT))

    while True:
        # start_time = time.time()
        rc, image = video.read()
        if type(image) == type(None):
            break

        image = cv2.resize(image, (WIDTH, HEIGHT))
        resultImage = image.copy()

        frameCounter = frameCounter + 1

```

```
carIDtoDelete = []
```

```
for carID in carTracker.keys():
```

```
    trackingQuality = carTracker[carID].update(image)
```

```
    if trackingQuality < 7:
```

```
        carIDtoDelete.append(carID)
```

```
for carID in carIDtoDelete:
```

```
    # print("Removing carID " + str(carID) + ' from list of trackers. ')
    # print("Removing carID " + str(carID) + ' previous location. ')
    # print("Removing carID " + str(carID) + ' current location. ')
    carTracker.pop(carID, None)
```

```
    carLocation1.pop(carID, None)
```

```
    carLocation2.pop(carID, None)
```

```
if not (frameCounter % 10):
```

```
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
```

```
    cars = carCascade.detectMultiScale(gray, 1.4, 14, 20, (24, 24)) #1.1 13 18
```

24

```
for (_x, _y, _w, _h) in cars:
```

```
    x = int(_x)
```

```
    y = int(_y)
```

```
    w = int(_w)
```

```
    h = int(_h)
```

```
    x_bar = x + 0.5 * w
```

```
    y_bar = y + 0.5 * h
```

```
    matchCarID = None
```

```

for carID in carTracker.keys():
    trackedPosition = carTracker[carID].get_position()

    t_x = int(trackedPosition.left())
    t_y = int(trackedPosition.top())
    t_w = int(trackedPosition.width())
    t_h = int(trackedPosition.height())

    t_x_bar = t_x + 0.5 * t_w
    t_y_bar = t_y + 0.5 * t_h

    if ((t_x <= x_bar <= (t_x + t_w)) and (t_y <= y_bar <= (t_y + t_h))
and (x <= t_x_bar <= (x + w)) and (y <= t_y_bar <= (y + h))):
        matchCarID = carID

if matchCarID is None:
    print(' Creating new tracker' + str(currentCarID))

    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

for carID in carTracker.keys():
    trackedPosition = carTracker[carID].get_position()

    t_x = int(trackedPosition.left())
    t_y = int(trackedPosition.top())
    t_w = int(trackedPosition.width())

```

```

t_h = int(trackedPosition.height())

cv2.rectangle(resultImage, (t_x, t_y), (t_x + t_w, t_y + t_h),
rectangleColor, 4)

cv2.putText(resultImage, str(carID),(t_x, t_y -15),
cv2.FONT_HERSHEY_SIMPLEX, 0.75, (200, 150, 250) ,2)
# TO Display id in the video

carLocation2[carID] = [t_x, t_y, t_w, t_h]

# end_time = time.time()

# if not (end_time == start_time):
#     fps = 1.0/(end_time - start_time)

for i in carLocation1.keys():
    if frameCounter % 1 == 0:
        [x1, y1, w1, h1] = carLocation1[i]
        [x2, y2, w2, h2] = carLocation2[i]

        carLocation1[i] = [x2, y2, w2, h2]

    if [x1, y1, w1, h1] != [x2, y2, w2, h2]:
        if (speed[i] == None or speed[i] == 0) and y1 >= 275 and y1 <= 285:
            speed[i] = estimateSpeed([x1, y1, w1, h1], [x1, y2, w2, h2])
            # speed = estimateSpeed([x1, y1, w1, h1], [x1, y2, w2, h2])
            if speed[i] > speedLimit:
                # print('CAR-ID : {} : {} kmph - OVERSPEED'.format(carID,
speed))

                print("overspeed", carID )
                saveCar(speed,image[t_y:t_y+t_h, t_x:t_x+t_w])
            # else:
            #     print('CAR-ID : {} : {} kmph'.format(carID, speed))

```



```

        # if speed[i] != None and y1 >= 180:
        if speed[i] != None:
            cv2.putText(resultImage, str(int(speed[i])) + "km/h", (int(x1 +
w1/2), int(y1-10)), cv2.FONT_HERSHEY_SIMPLEX, 0.75, (255, 255, 150), 2)

    # cv2.imshow('result1', image)
    cv2.imshow('result', resultImage)

    # out.write(resultImage)
    outputVideo.write(resultImage)





    if cv2.waitKey(1) == 27:
        break

cv2.destroyAllWindows()
# out.release()
outputVideo.release()

if __name__ == '__main__':
    trackMultipleObjects()

```

INFORMATION REGARDING STUDENT

Student name	Usn	Email id	Phone Number	Photograph
Rishav Surana	18BTRIS035	suranarishav08@gmail.com	9660353303	
Susanta Sarkar	18BTRIS048	sarkar.susanta18@gmail.com	9609736086	
Siddharth Kumar Harlalka	18BTRIS066	harlalkasiddharth789@gmail.com	9739969084	
Sujata Budha	18BTRIS065	suzatamst123@gmail.com	9148603973	

BATCH PHOTOGRAPH ALONG WITH GUIDE



Mr. Mathiyalagan R,
Assistant Professor,
Department of Information Science & Engineering



Mr. Siddharth Kumar Harlalka
18BTRIS066



Mr. Rishav Surana
18BTRIS035



Ms. Sujata Budha
18BTRIS065



Mr. Susanta Sarkar
18BTRIS048