`

Chapter 1

Introduction

In today's world, image processing concepts such as foreground extraction and key frame extraction are used extensively to reduce computational burden on post processing of the video. Post processing of the video can mean to extract features of objects of interest, for example face recognition or perhaps, for surveillance in cameras for analysis in detection of traffic violations, to name a few. In any case, one problem that many image processing techniques face today is variable frame rates and large amounts of background noise that is created during recording of the video being analyzed. This variable frame rate introduces lag in the video being analyzed and in turn results in the insertion of blank frames during frame extraction. Also, depending on the quality of the recording significant amount of noise can be present in the background causing loss of accuracy and in turn compromises the efficacy of the analysis. Thus, the variable frame rate and noise hinders meaningful frame extraction by resulting in frames that have no discernible foreground movement. Detection of motion of objects to apply to traffic applications that accomplish tasks such as violation detection, depend greatly on the quality of the video being processed. Therefore, an efficient pre-processing algorithm is imperative in order to ensure best results. In addition to this the method being used for detecting such a violation needs to benefit from the pre-processing stage or take full advantage of the improved quality obtained from pre-processing.

Much of the past and on-going research done in this field aims at resolving these issues in order to improve accuracy of results. There have already been various methods proposed to deal with filtering out unwanted frames that simply burden the computer with additional processing load whilst carrying no information of use during the analysis. The method that has become most popular is known as the Gaussian Mixture Model [1]. This method models each background pixel with a mixture of K Gaussian distributions, the value of K being three to five. The time that a pixel stays in the scene is determined by the weights of the mixtures in the distribution. The most probable background colours are the ones that stay longer as determined by the weights. In this method the number of Gaussian distributions 'K' has to be statistically determined through pre-processing of the video being analyzed. Therefore, the method suffers the disadvantage of

`

the time overhead required for this statistical analysis in order to obtain efficiency in results. Few more methods have been proposed since, notably the improved Gaussian Mixture method and the Bayesian Gaussian Mixture model to overcome the shortcomings of the first Gaussian Mixture Model method. The advantage in the first of the two is the adaptive choosing of the appropriate number of Gaussian distributions for modelling each pixel [2][3]. The second method incorporates Bayesian segmentation to target the specific issue pertaining to video analysis in variable lighting conditions [4].

However, all of these methods suffer from seemingly un-avoidable background noise. Also, they fail to address the issue of lag in frame rates of recording video thus, resulting in introduction of blank frames that, along with the foreground frames are unnecessarily processed during the analysis phase. Background noise can be decreased by frame averaging and blurring but this technique also decreases the quality of pixels forming the scene. In the proposed method , the concerns of variable frame rates and noise are addressed by making use of a clustering algorithm called 'K-means Clustering' to colour quantize the image using two clusters to obtain a clear de-noised image. This clustering is performed on the images obtained after background subtraction and thus frames that introduced as a result of the lag in frame rate are rendered completely black. Then the mean squared error similarity checker algorithm is used to filter out the blank frames and a video is compiled from the frames that contain significant foreground movement. Thus, the computational burden during the violation detection phase is greatly reduced. Furthermore methods that have been proposed to detect violation such as colour based tracking, mean-shift object tracking and Cam shift object tracking are dependent on local histogram distributions of pixels in the image [5][6]. Therefore, they inadvertently suffer the disadvantage of requiring a high quality video with perfectly preserved histogram distributions and no noise in order to track objects accurately. The proposed method implements a very straight forward and computationally highly economical method of violation detection, particularly signal jumps. It only requires basic geometric calculation and interfacing with a camera.

The research regarding de-noising, background subtraction for variable frame rate videos has long been a trending topic of interest among the image processing research community. As it is, various systems already exist to tackle the above mentioned problems, which are ubiquitous to all domains of further application. The following points introduce these systems.

`

- **K-SVD de-noising algorithm:** This method makes use of sparse and redundant representations over trained dictionaries, which represents the content of the image. Therefore, it uses Bayesian treatment of images to obtain de-noised images [8].

- **Adaptive wavelet thresholding:** It is also developed in a Bayesian framework and makes use of the generalized Gaussian distribution. The image is divided into wavelet sub-bands, and the thresholding process is adaptive to each sub band to accomplish de-noising of the image [9].

- **Global de-noising:** In this method a "Global Filter" is applied in which each pixel in the image is estimated from all other pixels in the image statistically, as opposed to applying adaptive filters to patches of the image [10].

- **Gaussian Mixture Model:** This method models each background pixel with a mixture of K Gaussian distributions, the value of K being three to five. The time that a pixel stays in the scene is determined by the weights of the mixtures in the distribution [1].

- **Improved Gaussian Mixture Model:** In this method the K distributions used for modelling is appropriately determined for each pixel in the image. Thus, it is more adaptive than the Gaussian Mixture Model [2][3].

- **GMM adaptive to variable lighting conditions:** This method incorporates per pixel Bayesian segmentation into the Gaussian Mixture Model in order to account for videos recorded in variable lighting conditions [4].

- **Adaptive variable frame rate coding:** This method adjusts the frame-rate of the video dynamically and adaptively, making use of information from already existing video encoders [11].

- **Intermittent motion coding:** This method involves disabling of motion coding during periods of inactivity in the video. Thus it records only parts of the video were active

- foreground movement is involved for further processing [12].

`

All of the methods explained above incur considerable overhead with regard to time or CPU usage. The Gaussian Mixture Model based methods cannot efficiently deal with variable frame rates in videos. The de-noising algorithms apply blurring techniques to the image which alter the quality of the image in question. The variable frame rate coding techniques make use of video encoder information, the compilation of which involves CPU overhead. Also, recording only during periods of activity means that the definition of activity in the scene has to be pre-determined in advance, and done so using extensive statistical analysis.

1.1 State of The art Developments

- The system achieves complete elimination of background noise
- The pre-processing phase ensures that the quality of the objects in the foreground modelling is almost as clear as the objects in the original frames.
- The method works better with recorded variable frame rate videos when compared to the standard used method, the Gaussian Mixture Model.

Many researchers have continued to innovate, contribute and increase performance of the image processing systems, since their inception. Innovation was revolutionized during the world wars when image processing and object tracking were indispensible tools with regard to air and naval warfare. Concepts that went into this endeavor have been improved upon with the advent of UAV's [7]. In more recent times, because of ever increasing traffic and population the need for efficient traffic management systems became a necessity. Image processing has found much application in this field. Many tracking algorithms, background subtraction methods and de-noising techniques have been proposed and implemented to help with the pre-processing of traffic scene recordings, which is an unavoidable step. Although, it is still not perfect, significant progress has been achieved with regard to improving quality of footage, removing noise and frame rate lags, removing shadows and illumination changes, etc. The proposed work aims to further this research and apply it in detecting violations such as signal jumps in a computationally economic, yet accurate way.

Employing image processing techniques to achieve improved quality of traffic scene recordings to facilitate violation detection presents a daunting, yet very interesting challenge. These challenges include dealing with background noise, variable frame rates, illumination

`

changes and shadow removal. Also, the need for removal of certain parts of the video that can add no value to the analysis of the scene is vital with regard to decreasing computational burden. The following explains these issues in brief.

1.1.1 De-noising

Noise refers to minute disturbances in the pixel shading of an image. The recording on a camera cannot exactly simulate how the scene looks in reality. There is always present background noise occurring because of the internal operation of a camera. The optics involved cannot eliminate all the noise. In still pictures, this is a simpler problem. However, in highly dynamic video recordings such as traffic scenes, de-noising the footage is a very challenging task.

1.1.2 Foreground extraction

Foreground extraction refers to the separating of moving objects in the scene from background objects, including non-static background objects. For example, a tree slightly displaced by the motion of a wind current in the scene is still considered a background object. Defining what can be categorized as a foreground object in a dynamic scene is a delicate problem and is also highly application specific.

1.1.3 Variable frame rate

Videos when recorded have a feature associated with them called frame rate or "frame rates per second (fps)". The video can be recorded at 10fps or 25fps, whatever is the case, and the algorithm used for processing of the video must work in both cases. Also, the lag due to shifting frames introduces many unwanted frames that become evident during extraction and can bring down the computational efficiency of the task being carried out.

1.2 Motivation

When trying to develop a system that can detect signal jumps by vehicles in a traffic scene, firstly, the video recording of the scene needs to be free of background noise. Moreover, an efficient background modelling technique is required in order to obtain an accurate foreground mask, while also ensuring that non-static background objects are not recorded in the foreground mask. Also, when footage is recorded, the frame rates per second introduce lag in the

`

video and thus results in insertion of unwanted frames that slow down the CPU during analysis of the traffic scene for violation detection.

After all this is accomplished, already existing methods such as mean-shift and cam shift can be used effectively for object tracking. However, these techniques make use of feature extraction methods such as SURF which are highly CPU intensive tasks [13]. The widely popular Haar Cascade Vehicle Classifier can also be used to track vehicles in the scene and obtain characteristics [14]. The problem is that these techniques are computationally highly complex and burdensome. Therefore, a new system has been developed, which makes use of simple geometric calculations to detect signal jumps in traffic scenes.

1.3 Problem Statement

The system works to achieve key frame extraction, which is done based on the movements of objects, in videos with variable frame rate followed by detection of traffic violation. The nature of the traffic violation considered in this project is signal jump. The video in the problem is recorded in a real life traffic scene.

1.4 Objectives

- The objectives of the system developed are to:
- De-noising of the recorded traffic scene video,
- Enhancing the quality of the video recorded,
- Eliminating blank and redundant frames,
- Detection of vehicle violation,
- And Vehicle Identification.

`

1.5 Scope

The system takes a MPEG-4 or AVI format video recording of a real life traffic scene, performs background modelling for inclusion of non-static background images. The system performs pre-processing to enhance the quality of the recorded video and then proceeds to detection of vehicles responsible for violation in the video. It also performs extraction and filtering of the violating vehicles to identify the vehicle.

1.6 Methodology

A video of a traffic scene is recorded using a standard resolution camera. The video is given as input to the system. The background in the traffic scene is modelled using the running average method [15][16]. By assuming a reasonably high regulator value, non-static background objects is successfully incorporated into the background. Background subtraction of all the frames is carried out against this background. The standard frame differencing for background subtraction has been used. Other methods can also be used to the same effect [17].

Next, the foreground masks obtained are clustered using K-means Clustering with two clusters thus obtaining two coloured colour quantized images of the foreground. The foreground images so obtained contain no noise and are as clear as the original frame [18]. After colour quantizing the images, the frames responsible for the lag are rendered completely blank. Hence, we make use of a simple Mean Squared Error similarity algorithm to filter out these blank frames and recompile the video [19].

From the newly compiled video, the images are contoured and then the largest contours in the image are identified [20]. Thus, this contour represents the vehicle of interest. The contour is then approximated into the shape of a polygon whose centroid can be calculated [21]. The distance from the centroid to the edge of the frame is calculated. When the distance is less than an assumed threshold, the frame of occurrence of violation is captured and time stamped.

Next, the back view of the vehicle is captured using a electric signal to a secondary camera. Subsequently, the plates are extracted and numbers are identified using a digit classifier trained using support vector machines. The numbers are identified using histogram of oriented

`

gradients [22]. Following this step information is gathered from the database indexed by the identified numbers are the filtered results are displayed.

1.7 Organisation of Thesis

This report contains nine chapters which are described briefly as follows:

Chapter 2: This chapter describes the fundamental concepts of Background Subtraction, K-means Clustering and Image Contouring required to understand the proposed system.

Chapter 3: This chapter describes the hardware and software requirements that are necessary for proper execution of the system.

Chapter 4: This chapter describes the High level Design of the project and the flow of data between individual modules using Data Flow Diagrams.

Chapter 5: This chapter describes the detailed design of each of the modules that are integrated to form the system. This is illustrated using a structure chart of the system and flowchart describing the control flow through the various modules.

Chapter 6: This chapter describes the required software setup and environment required for successful implementation of the system.

Chapter 7: This chapter describes the detailed testing of the individual modules, the integrated system as a whole and the system as a whole with various input scenarios.

Chapter 8: This chapter describes the results obtained from the testing process and illustrates the results in a tabulated form.

Chapter 9: This chapter describes the conclusion of the proposed system and provides insight into future enhancements that can be incorporated to correct the systems limitations.

1.8 Summary

This chapter introduces briefly the system being developed, the current standards for implementation of traffic systems. It also specifies the shortcomings and limitations of existing models of use, providing justification for the need for development of a new system. Finally, it briefly outlines the methodology used to develop the system.

`

Chapter 2

Fundamental Concepts of Background Subtraction, K-means Clustering and Image Contouring

For any application requiring Image processing of video, background subtraction and modeling, clustering, checking the similarity of images, de-noising and contouring are all very basic pre-requisites. This chapter gives an overview of all these concepts that will help elucidate the design and implementation details covered in the chapters that follow.

2.1 Background Modeling

When video is captured in real life scenes and objects in the scenes are analyzed, there are two types of objects. Background objects and foreground objects. For example, consider a typical traffic scene. Vehicles, pedestrians and maybe even some birds are all classified as objects that move. The objects such as buildings, trees are all stationary. Therefore they are considered background objects. While this may seem quite simplistic at first, it is not so straight forward. Now, imagine that there are people in the background that are talking using hand gestures or that there are tree branches, the position of which has been displaced by a gush of wind. Even though these objects are strictly speaking moving objects, they are considered non-static background objects. Therefore, foreground objects are objects of interest with respect to a specific application, those that move a significant amount against a modeled background. Background Modeling refers to the process of constructing this background. There are several ways to accomplish this. The running average method has been used, where every frame is averaged at a rate determined by a regulator value. If that value is equal to a certain threshold, that threshold being decided by the domain of application, the non-static background objects are also included in the background modeled [15][16]. There are other methods of background modeling such as the Gaussian Mixture Model method [23]. There are also lesser known but reasonably efficient methods such as the codebook construction method [24]. However, as stated above the goal is to achieve efficiency while maintaining simplicity. Hence, a statistical approach that is computationally economical and efficient has been adopted.

`

2.2 Background Subtraction

Background Subtraction refers to the method of literally just subtracting all the frames constituting a video against a modeled background of the scene. Thus, all the moving objects or foreground objects are separated out and the result appears as a black background with white objects moving against it. There are various techniques that can be used to carry out background subtraction [17]. One very popular method is known as the Gaussian Mixture model which uses a mixture of K Gaussian distributions to model the foreground [1]. The result looks as shown in figure 2.1.

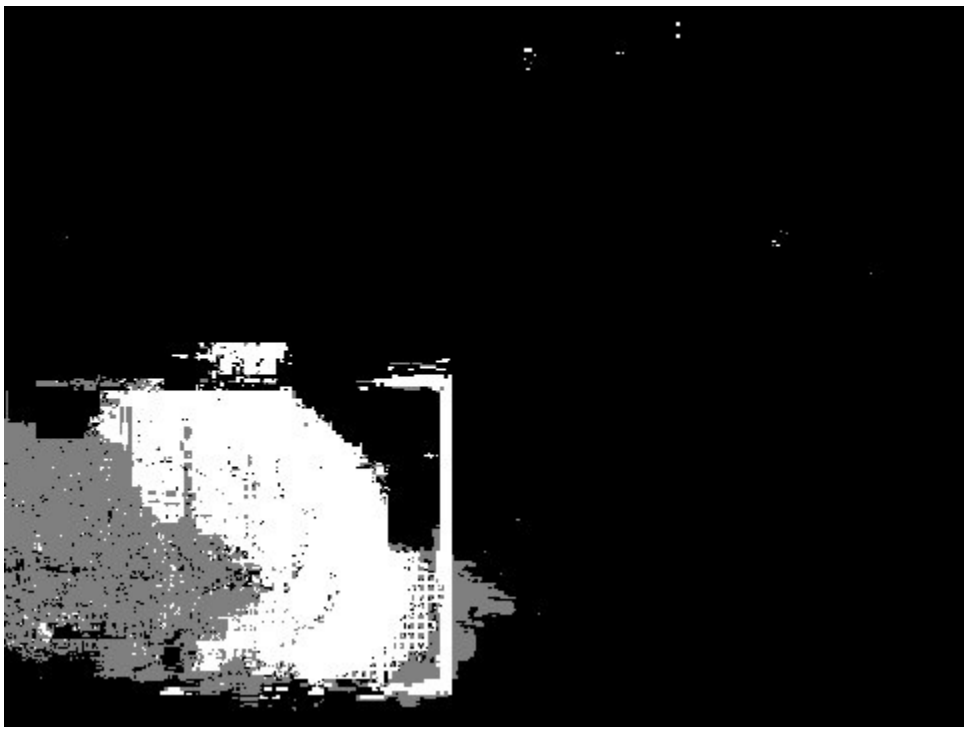

Figure 2.1: Background subtracted image using GMM

However, as is evident from the figure the Gaussian Mixture Model approach suffers from lack of detail and background noise. The proposed approach resolves both these shortcomings by using clustering and de-noising techniques.

2.3 K-Means Clustering

K –Means Clustering refers to a method of clustering a dataset based on the Euclidean distance when the dataset is mapped on to a co-ordinate system. For example, consider that a

`

group of 10 people need to be grouped into two groups or clusters as per their height. First all the 10 people are plotted on the XY plane against their height. Next, two random points with maximum separation are chosen as the centroids of the two clusters (since we need all the points to be grouped into two clusters). Of course, there are improved versions of the algorithm where the number of clusters is decided adaptively specific to the application [25][26]. But this feature is not required for the grouping of 10 people and hence the number of clusters is set as two for this example. Another point among the eight remaining points is chosen. It is grouped along with the cluster that it is closest to. Subsequently, for that cluster, the new average is calculated by averaging the newly added point with the old one. In this manner, all the points are grouped into two clusters thus accomplishing our demarcation based on height.

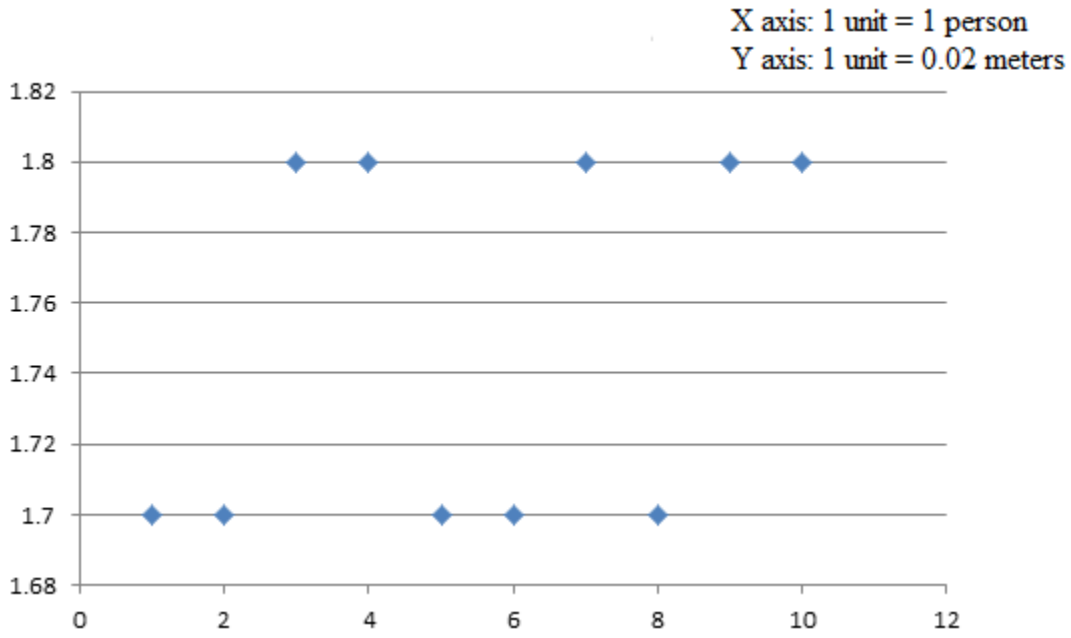

Figure 2.2: Height of 10 boys in meters before grouping

`

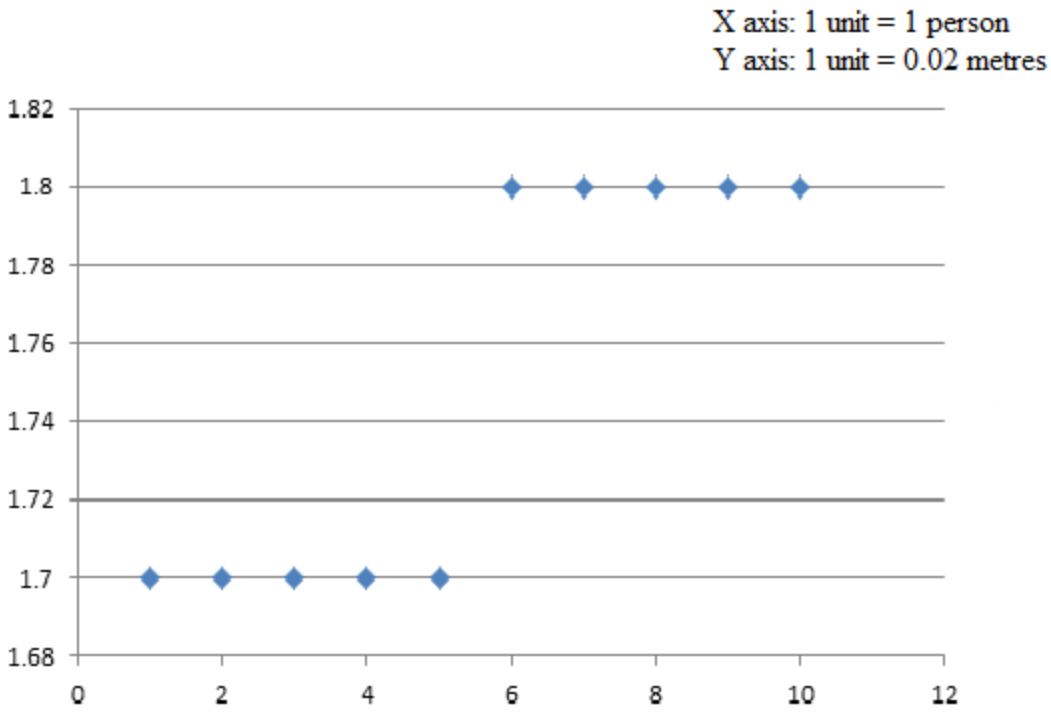

Figure 2.3: 10 Boys grouped in two clusters according to height

Figure 2.2 illustrates points before clustering and figure 2.3 illustrates points after clustering. X represents the person number and Y the height in meters.(refer the figures after it)

2.4 Image Contouring

In image processing objects need to be identified in the most computationally efficient way because image processing in general is highly computer intensive; therefore, it is beneficial to accomplish certain functions in less computer intensive ways whenever possible. For example, consider that in an image of still coins of different sizes, the size being representative of their values; it is required to find the coin with the median value. In statistics, median means the middle value in the sorted set of values. If we were to perform this action using standard image processing techniques, the image can be thresholded, the gradients of the image elements can be found and compared to the shapes of numbers identified using a digit classifier. Another way of doing it would be to compare the coin images with a database of stored coin images using a structural similarity index. A third way would be to compare color histograms of areas in the image to those already obtained before we start processing. However, while all these tasks present a definite way to obtaining the desired result, they all involve complex image processing.

`

Now, if the same could be accomplished with a much simpler technique it would solve computational efficiency. Introducing Image contouring; it is the process of identifying contours or outlines of objects in an image. So in the coin problem if a set of contours could be identified, sorted by area after approximating them into standard polygonal shapes and then the middle term in the list can be picked out. Likewise, Image contouring can be used to simplify a lot of problems in real life as well. In this method for example, contouring has been used to find the vehicle of interest in the frame. In a frame sized window, the object corresponding to the largest contour is most likely to be the vehicle. This is a reasonably safe assumption. Moreover, this method is highly computationally simple and economical.

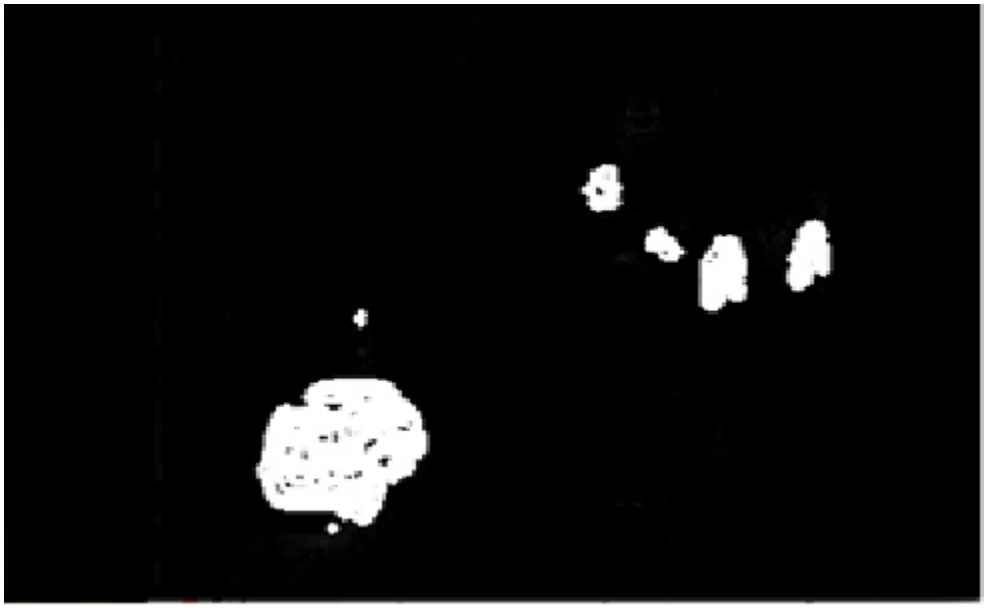

Figure 2.3: Contoured image of car and pedestrians.

Figure 2.3 shows a contoured image of a car and few pedestrians.

2.5 Summary

In this chapter, the basic concepts required for understanding the details of implementation of the project and the mathematics behind it has been briefly explained. In order to understand how these techniques are relatively advantageous, a background in basic image processing concepts used in this project is necessary. Also, K- means Clustering has been explained with respect to a random data set involving the heights of persons.

`

Chapter 3

Software Requirement Specification

Along with the functions involved, it is necessary to choose the right software support that can provide libraries that help facilitate the implementation of the various modules incorporated in the project [27]. Also, the characteristics and user interaction interface is instrumental in facilitating the ease of use of the system. Moreover, the software must be chosen such that it is platform independent and runs seamlessly across all platforms.

3.1 Overall Description

This project is developed in the python language version 2.7 and is also scalable to version 3.4 and above. It makes use of the open CV 2 series library that contains significant changes from the open CV library. All the functions are self contained and are not dependent upon the platform using which the system was developed.

It consists of four modules. The background modeling phase, the foreground extraction (background subtraction) phase, the clustering and filtering phase and finally violation detection followed by plate number digit recognition. The filtered information extracted is then retrieved from a simulated database. All of the functions have been implemented using python standard libraries and no additional software other than python 2.7 and above and open CV 2 is required to implement the system.

3.1.1 Product Perspective

The system is mainly used to achieve red signal jump detection following some computationally efficient pre-processing techniques. The simplicity of approach and accuracy of results make it a tantalizing prospect for future work. The system integrates between all the modules seamlessly and produces good results in various tested scenarios [28].

The system is a stand-alone application that requires only that the standard python and openCV libraries be installed on the platform running the software. The Graphical User Interface, to avoid third party intervention is also modeled using openCV’s drawing functions

`

and basic terminal commands. The GUI is simple in design and is extremely easy to use for a lay-person that needs the system to meet his own ends.

3.1.2 Product Functions

The application accepts recorded video of MPEG-4 or AVI formats, models the background based on the scene, performs background subtraction of the video frames against the modeled background, Color quantizes the resulting frames using K means Clustering (K equals 2), uses a mean squared error similarity checker to filter out blank frames and compiles the remaining frames into a new video. The new video is processed to detect violation.

Therefore the system models the background, performs background subtraction, extracts full colored and foreground key frames, simulates signal time, detects signal jump and even recognizes number plate digits using a trained classifier. In addition to this it also filters and retrieves results from a simulated database. The system performs all these operations both in a modular and an integrated fashion depending on user interaction with the Graphical User Interface.

3.1.3 User Characteristics

It is important to educate oneself about the type of users that will typically require such a system for their specific needs. Knowledge about the characteristics of the users can help design an easy to use, efficient system and also, help design user specific Graphical User Interface components. The system is intended to be used as per user specification as there is very little broad scope of usage of such an intricately designed application. Therefore, knowledge about the user using the system is required to tailor the system to their needs.

3.1.4 General Constraints

The performance of the system is subject to the following constraints:

- The level of vehicle occlusion in the video has to be very limited.
- The proximity of shooting of traffic footage needs to be sufficiently high. Small and blurry videos of vehicles at a distance will not work with the system.

`

- The system camera must be placed laterally by the side of the signal post to capture video feed perpendicular to the post.
- The system camera must be placed in line with the post, as the edge of the frame in the video is assumed to be the real life position of the signal post displaced by an appropriate distance.

3.1.5 Assumptions and Dependencies

The assumptions made by the system are that the camera is placed in line with the post for the edge of the frame to qualify as the real life position of the signal post. Also, the system can interface electrically with a high definition camera placed on top of the signal post perpendicular to the primary camera in order to capture HD still images of the vehicle responsible for signal jump, once it crosses the red signal. This is done to accurately extract plate information. Finally, it is assumed that the system runs only when the light in the signal is red as it makes no sense to perform all the computations for violation detection when the signal is green.

3.2 Specific Requirements

This section describes the functional requirements of the system [29]. This outlines all of the working functionality that the system is intended to provide. The minimum hardware and software configuration required to run the system without lag is also described. Also, performance benchmarks as per which the system is expected to comply is detailed. Finally, the interfaces required to run the system and the non functional requirements is described [30].

3.2.1 Functional Requirements

Functional requirements describe the function of each of the individual units or modules that the system is composed of. Also, it describes the functionality that the integrated system as a whole, is expected to provide [29].

The following functional requirements are to be met by the system:

- Background modeling of the recorded scene

`

- Identification of non-static background objects, to be incorporated into the modeled background.
- Frame differencing of the video frames against the modeled background or background subtraction.
- Elimination of all background noise in the video.
- Estimation of the time required to run the system or estimation of the simulated red signal time.
- Reducing the number of frames required for post processing, in this case violation detection caused as a result of the frame rate of the recorded video.
- Detailing the foreground modeling of the scene to accurately depict the features and characteristics of the moving objects.
- Detection of jumping of a red signal by a moving vehicle in the scene, both four and two wheelers.
- Capturing the instance of the video, when the violation occurs.
- Time stamping the instance of the video so captured, for user understanding and information.
- Capturing the image of the vehicle performing the violation act and extracting the number plate.
- Recognizing the numbers of the number plate and retrieving information about the vehicle performing the violation, from the database, by indexing the records by the extracted numbers.
- It must be able to handle errors in input given by the user and provide necessary output messages as corrections.
- It must provide a simple, yet interactive Graphical User Interface that is intuitive and works efficiently when used by a lay man.

3.2.2 Performance Requirement

The system must operate smoothly when under use by a single user, either on the command prompt or Graphical User Interface. Simultaneous working on both must also yield the same result. It must work the same for all acceptable formats of input video recordings and provide consistent and reliable outputs. The time taken for execution must be commensurate with

`

the size of the input being processed. The results must comply with the standards and must be acceptable to every specific type of user.

3.2.3 Software Requirements

The software required for development of the automatic parallelization tool are:

Operating System	:	Windows, Linux or Mac
IDE/Workbench	:	vim editor or pyCharm
Programming language	:	Python2.7 or higher
Other Required Software	:	OpenCV version 2.0 or higher.

3.2.4 Hardware Requirements

The hardware required for the project are (minimum requirements):

Camera		Standard webcam 2.0 MegaPixels
Processor	:	Intel Core 2 duo or higher
Processor Speed	:	1.2 GHz or higher
RAM	:	1 GB or more
Hard Disk	:	100MB of free space

3.2.5 Design Constraints

The system requires that the camera be positioned at angle suitable for recording the side view of the vehicles moving the scene. Also, all of the standard python and openCV libraries required for running the software are needed.

- The design must be applicable to both low resolution and high resolution videos.
- The input video can be shot indoor or outdoor.
- The input video can be in MPEG-4, AVI or MOV video formats.
- The moving object detection should be completed in finite time interval.
- The background image must be displayed alongside

`

3.2.6 Interfaces

The system provides a terminal or command line interface for execution of the program both as individual modules and as an integrated unit. It also interfaces effectively with files and databases necessary for storage and retrieval of information during user interaction. It also provides an intuitive Graphical User Interface for users that have limited knowledge of the command line interface.

3.2.7 Non Functional Requirement

These requirements describe integrity and security requirements of the system necessary to ensure accountability towards the end user and overall efficiency of the system [30].

- **Reliability**: The system must not fail or crash under any circumstance of use. It must provide core functionality for all inputs subject to the requirements in terms of format and other input constraints. The output obtained must be consistent and reliable throughout.
- **Availability**: The system must be available for the end user, availability being defined by the user's jurisdiction and discretion. In other words, the system must never be unavailable for use when the user needs it.
- **Security**: The system must ensure secure functioning of all the individual units and integrity of information being stored and retrieved from virtual and intangible sources such as files and databases.
- **Maintainability**: The system must allow for bug fixes, if identified by the user.
- **Portability**: The system must be executable in different operating system environments.

3.3 Summary

All the requirements both functional and non functional of the system have been described in this chapter. It also provides detailed description of the minimum hardware and software requirements essential for execution. It also outlines the constraints, assumptions and dependencies upon which the system was built. Lastly, it describes the interfaces, modules and the perspective with regard to which the system was designed.

Chapter 4

`

High Level Design for Object tracking for video analysis in traffic scenes

This chapter describes and illustrates the design of the system used to achieve foreground modeling and subsequent signal jump detection and analysis. The different modules in the system and data flow between them are also described. The approach used in each of the modules is also briefly described.

4.1 System Architecture

Large and complex systems need to be decomposed into smaller subsystems for effective co-ordination and efficiency. This traffic violation detection system consists of many modules that constantly interact with each other as well as sources of information in the form of databases and files. Therefore, a modular design approach to the system is imperative [31][32].

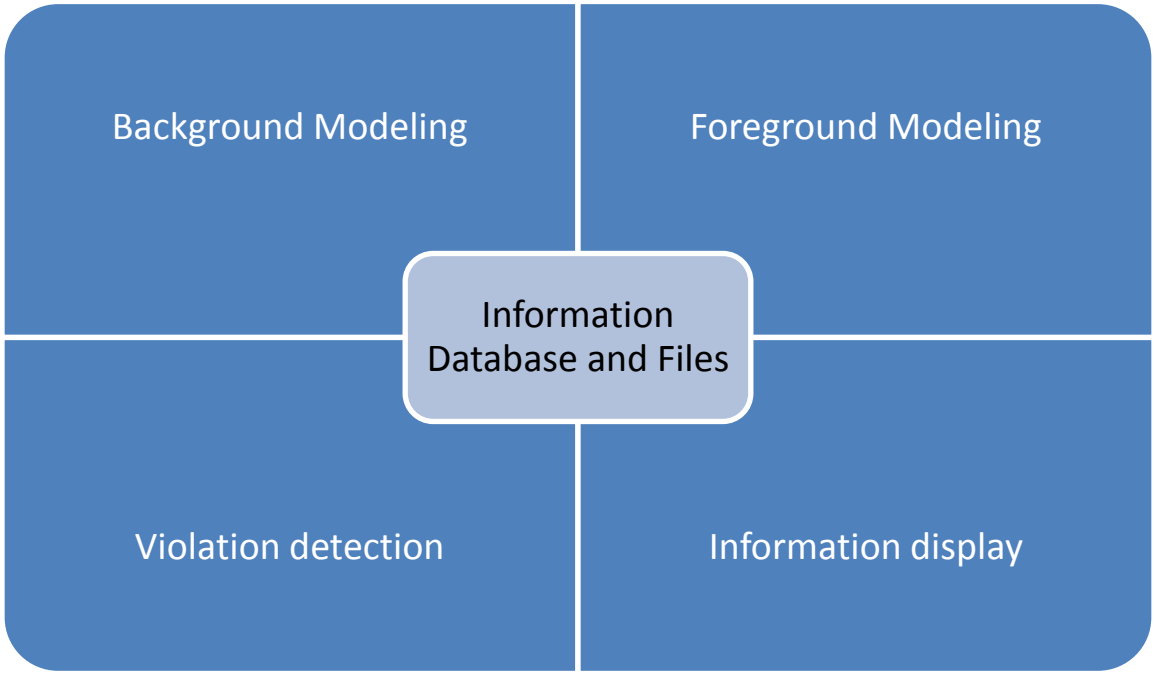

Fig 4.1: System Architecture of Violation detection System.

The system architecture of the violation detection system is as shown in Fig 4.1. The input is given to the background modeling phase and the output is obtained from the information display phase. The system is divided into four modules, all interacting with an information

`

database and files for storage and retrieval. The four modules are in order of execution, background modeling, foreground modeling, violation detection and information display.

- The background modeling is carried out by a frame averaging technique known as the running average method, in which each frame is averaged against the previous frame. Therefore, the intensity of the pixels that form the foreground is gradually decreased to a point of disappearing from the scene. Thus, the background is modeled. The rate of averaging determines the amount of non-static movement that the system accounts for.
- The foreground modeling phase further includes three phases:
    - Background Subtraction: The frames are subtracted against the modeled background to obtain moving objects in the scene.
    - K- Means clustering quantization and filtering: The background subtracted frames are color quantized using two clusters and the blank frames are filtered out using a mean squared error similarity checker.
    - Recompilation: The frames obtained as a result of clustering contain no noise and are recompiled into a video for the next phase of processing.
- The violation detection phase contours and approximates the contours corresponding to the foreground objects. The distance from the centroid of the largest contour to the edge of the frame determines the instance of violation of the object. This instance is then captured and time-stamped.
- The information display phase: Obtains information from the plates using machine learning and retrieves information from the interacting files and database upon request.

Finally, the results of the display phase are obtained as output from the system.

4.2 Data Flow Diagram

A data flow diagram (DFD) illustrates how data is processed and flows through a system in terms of its input and outputs. Hence it is the graphical representation of this flow of data. There are several formal representations of Data flow diagrams, all of which describe only the flow of data [33]. For complex systems DFD's often need to be decomposed into sub-levels for a clearer depiction of the data flow through the various modules of the system [34].

`

4.2.1 DFD Level 0 – Object tracking for video analysis in traffic scenes

The most basic level is the level 0 which indicates the interaction of the end user with the Object tracking for video analysis in traffic scenes system which is the system being discussed.

As shown in Fig 4.2, the user being the Source sends the recording of the traffic scene as input to the system. The system gives the details of all the violations that are detected in the recorded scene as extracted from the database.

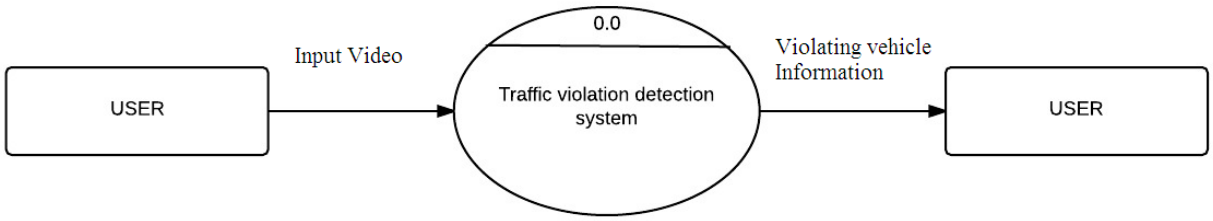

Fig 4.2: DFD Level 0 – Object tracking for video analysis in traffic scenes

4.2.2 DFD Level 1 – Object tracking for video analysis in traffic scenes

The next higher level of the DFD is Level 1. This level gives a clearer view of data flow than that of the previous level, 0 [34]. The module Traffic Violation Detection System from the Level 0.0 of the DFD is expanded into 5 sub-processes that define the work of the system in a modular fashion.

As shown in the Fig 4.3, the first module is the Background Modeling that takes the recorded traffic scene, as the input and models the scene background. .

`

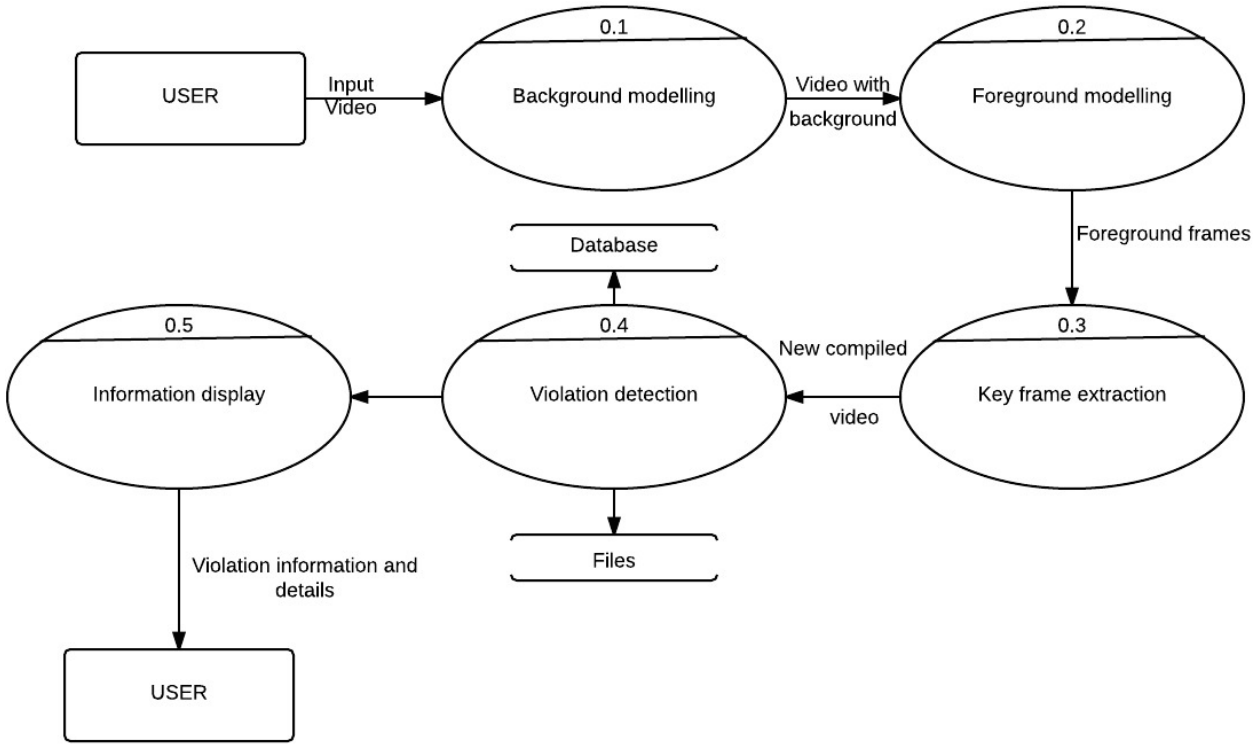

Fig 4.3: DFD Level 1 – Object tracking for video analysis in traffic scenes.

The second module, foreground modeling, gets the modeled scene background from the background modeling phase and subtracts all the video frames of the recorded traffic scene initially given as input, against the modeled scene background in order to obtain only the foreground objects or objects that show significant movement between successive frames.

The third module, Key frame extraction, gets the foreground frames from the previous stage, eliminates redundant and blank frames, compiles the new frames into another video and passes it on to the next stage of processing.

The fourth module, Violation detection, detects instances of violation and stores details of the vehicle objects participating in the violation in the database and files.

The last module, the information display phase displays information from the files to the end user.

4.2.3 DFD Level 2 – Foreground Modeling

The Foreground Modeling module from Level 1 DFD is further expanded in the DFD Level 2. It is as shown in the Fig 4.4. The subdivided module Background Subtraction takes the

`

input video with the modeled scene background from the background modeling stage. It performs background subtraction of video frames against the modeled background to obtain foreground frames. These foreground frames are taken by the next submodule to cluster the foreground frames to eliminate background noise and enhance the quality and clarity of the frames.

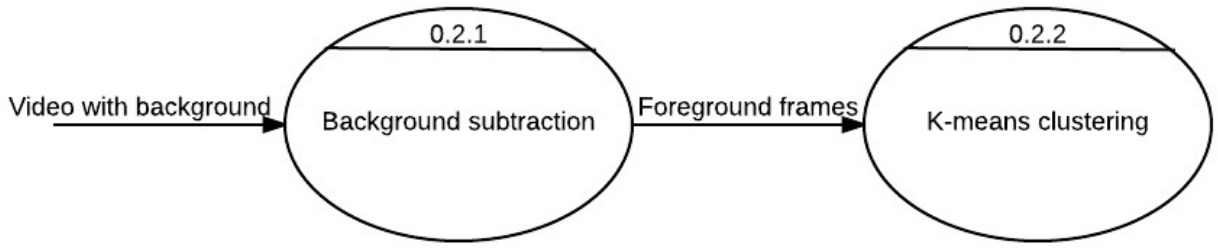

Fig 4.4: DFD Level 2 – Foreground modeling.

4.2.4 DFD Level 2 – Key frame extraction

The Key frame extraction module of the Level 1 DFD is exploded further in the DFD Level 2. The subdivided modules are shown in Fig 4.5. The Mean squared error checker submodule takes the foreground frames and compares all the clustered frames to a blank frame, thus filtering out all the blank and redundant frames in the video, resulting in Key frames. The Re-compile key frames into video submodule takes the Key frames from the previous submodule and makes a new video from the Key frames to be used for the next phase of processing.

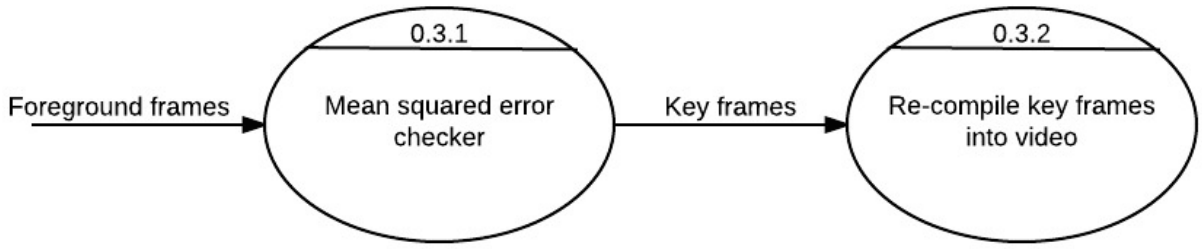

Fig 4.5: DFD Level 2 – Key frame extraction.

`

4.3 Summary

The design method followed in the development of the system is outlined in the high level design. System architecture shows the various sub-systems that make up the system as well as the interactions and flow of data and information between them with illustrated diagrams to show the same.

Finally, the data flow diagrams that represent strictly only the flow of data between various stages of processing, from input to output are illustrated and in some cases, further decomposed to detail the data flow. The Data Flow Diagrams are represented in various levels including DFD level 0 (context), DFD Level 1 and DFD level 2.

`

Chapter 5

Detailed Design of Object tracking for video analysis in traffic scenes

Various modules that perform specific functions are integrated to form the system as a whole. The first section describes the overall structure of the system and interfaces between the different modules. The functional descriptions of the modules are explained in the next section.

5.1 Structure Chart

A structure chart shows the organization of system into the different modules integrated to form the system [44]. It illustrates the practice of writing programs in a structured fashion involving modules that interact with one another efficiently. It consists of different modules represented as rectangles and data or state flow between them as an arrow with a circle. Hence, interactions between different modules of the system can be represented using structure diagram.

The core functions of this project can be divided into following modules:

1. Learn Background.
2. Extract frames.
3. Background Subtraction.
4. K-means Clustering.
5. Key frame extraction.
6. Violation Detection.
7. Finding Contours.
8. Determine Largest Contour.
9. Vehicle Identification Information extraction and display

`

The interactions and control flow between these 9 modules are shown in Fig 5.1. Detailed explanation of these modules are given in the next section.

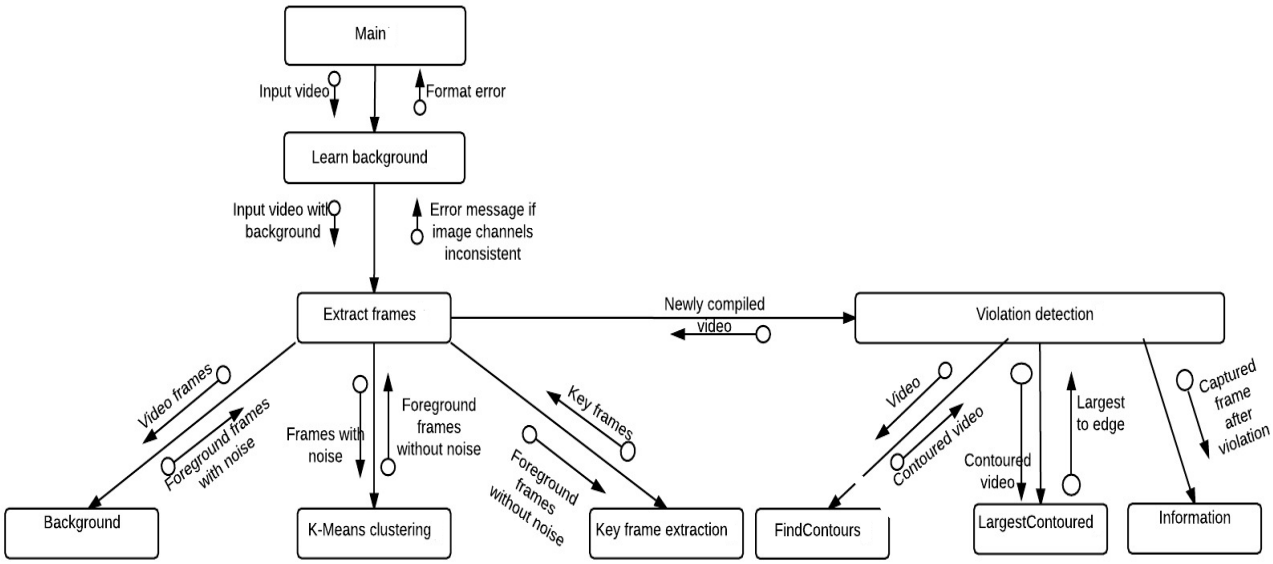

Fig 5.1: Structure Chart

As shown in Fig 5.1, the main function invokes the learn background module and passes it the input video, in case of error encountered in format the module returns the error. After the background is modeled it is passed to the extract frames module which performs background subtraction, followed by K-Means Clustering of the frames and finally key frame extraction to result in a reduced newly compiled video. The compiled video is passed through the violation detection phase, which finds the largest contour representing the vehicle object in the frame and then extracts vehicle identification information from the database and files.

5.2 Functional Description of Modules

The functional description of modules shown in the above figures and interaction between different modules are given in the following section.

`

5.2.1 Learn Background

This module performs extraction of the frames of the video and averages the frames at a rate determined by a regulator value. The goal is to obtain the background contained in the scene from the recorded traffic video without any of the moving vehicles. This includes cars and other vehicles. However, there can be slight movements in the background known as non-static background movements, for example, the rustling of trees and branches, or gesture motions of people making conversation in the background. Such motions have to be included in the background so that they do not show up as foreground objects in the modeled foreground. Consider equation (1),

F_n is the frame obtained after averaging frame F_0 and the previous F_n at a rate determined by the value of r . It is called the regulator value. If the value is chosen suitably, the non-static background images can be incorporated into the modeled scene background.

$$F_n = (1 - r)F_n + rF_0 \quad (1)$$

5.2.2 Extract frames of the video

This module extracts the frames of the video to facilitate the subtraction of the frames against the background in the background subtraction stage.

5.2.3 Background Subtraction

This module subtracts all the frames of the recorded traffic scene against the scene background modeled in that particular traffic scene. Following this step all objects that are in motion against the background are modeled. However, some background noise also escapes and forms a part of the foreground model. Also, because of the dynamic nature of the background subtraction process, the clarity of the foreground modeled is somewhat compromised and hence a post processing of the foreground modeling is required.

5.3.4 K-Means Clustering

This module clusters the pixel intensities into one of two colors. This process is called color quantization of frames [18]. After, color quantization the foreground modeled becomes

`

abundantly clearer and very closely resembles the object in the original video frame before the background subtraction. Also, as a result of grouping pixel intensities into one of two clusters, all the background noise generated by the background subtraction stage is eliminated. Therefore, the clustering ensures that all of the background noise generated is completely eliminated.

5.3.5 Key frame extraction

In this module, the frames resulting from clustering are compared against a blank frame to determine a similarity index. The algorithm used for the comparison is known as the mean square error similarity checker algorithm [19]. In this method, the difference between the pixel intensities of the image and the mean pixel intensity are squared and cumulated to form a sum. This sum is then divided by the total number of pixel's representing the pixel intensities in the image. Therefore, if the result of the mean squared error for the pixel intensities of two frames is close to zero, the frames are considered very similar. In this project, the threshold considered for similarity is 0.05 to account for the margin of error. Consider the formula in equation(2),

$$\frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[I(i,j)-K(i,j)]^2 \quad (2)$$

m, n, represents the total number of pixels in both the frames,

i, j, represent the pixel in consideration being differenced with the mean and squared.

The result of the operation, in case of similar frames is always a number close to zero. Therefore, when the clustered frames are compared to the blank frames using this technique, if they are found similar, they are eliminated. Also, if any of the subsequent frames are found similar to the previous frames, they are also eliminated to remove redundancy.

Finally, the original frames corresponding to the key frames obtained from the key frame extraction process are recompiled into a MPEG-4 format recording with the same bit rate and passed on to the next phase for the actual violation detection to take place.

5.3.6 Violation Detection

After the new video is compiled, the background subtraction takes place once again. The frames of the new video are subtracted against the background modeled initially, after which the

`

resulting foreground model of the frames are contoured after the application of a Gaussian filter to smooth out the edges. Contouring is the process of outlining the boundaries of the objects in the frame [20]. Once, the contouring is carried out, in the scene, the vehicle object is always the object with the largest outer boundary. Hence, the contour with the largest area is found using a built in openCV library method and stored in an array. This is done because images are represented as numerical python array's in pythons openCV implementation. Contours can also be approximated into standard geometric shapes to make calculations easier. This is done on the contour with the max area and the center of the shape is found. In this project, we approximate it to a rectangular shape. The distance from this center to the edge of the frame is calculated. If this distance falls below a particular threshold, the threshold is considered as 30 pixel units in this project, the vehicle has performed a violation. At this instance, the frame is captured and time stamped. Two frame instances are captured; the frame before the violation and the frame after it.

5.3.7 Finding contours and identifying largest contour

This module is responsible for identifying the outlines or contours of objects in the foreground model. This process is called contouring. Finding image contours first requires applying a bilateral filter or Gaussian filter to the frame [45][46]. Next, the edges in the frame are found using the canny edge detector [47]. The edges are then outlined using the contouring process. The largest contour among all that are identified in the frame is calculated using top down segmentation based on the area of the contours [48].

5.3.8 Vehicle Identification

In this module when the vehicle performing the violation is identified, the number plate is extracted from the vehicle image and the numbers on the plate are identified using a combination of support vector machines and histogram of oriented gradients [38][39]. The database is indexed using the identified plate numbers and the information is displayed to the user upon request.

`

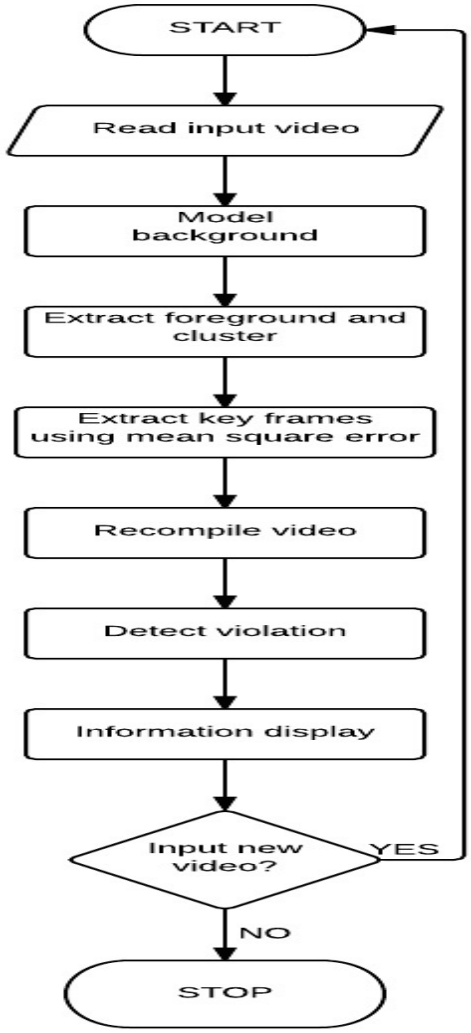

Fig 5.2: System Flowchart

The flowchart for the entire process of Vehicle Violation Detection is outlined in Figure 5.2

5.4 Summary

This chapter describes the detailed design of the project. The problem given is sub-divided into different modules. Structure Chart is constructed and modules used in the system are identified. In the first section the overview of the structure of the project is presented. Following this section the description of functions of the modules of the system is given. The modular nature of the project is shown in this chapter.

`

Chapter 6

Implementation of object tracking for video analysis in traffic scenes

Software is considered to be a deliverable once its design is transformed into executable codes on the platform on which it is intended for use with requirements adhering to best available coding standards and practices. Implementation phase of software consists of important decisions regarding selection of platform, programming language to be used, coding conventions to be followed etc. which are influenced by several factors such as the environment in which the system works, security requirements and issues, optimization concerns, efficiency and accuracy of the system and its results.

6.1 Programming Language Selection

The efficiency of the system is dependent on the programing language used to implement the software. Also, it depends on the need to implement the system on any platform required by the intended user. It also decides to what extent the language chosen can take advantage of efficient image processing libraries. Python version 2.7 and above has been chosen for developing this system.

The OpenCV library is widely supported by many versions of python [35]. Also, standard libraries such as the scikit machine learning library required for image processing are very efficiently implemented in the python programming language.

6.2 Platform Selection

The platform selected for developing the system is the Mac OSX version Mavericks because the setting up of the python environment and openCV libraries is very simple owing to a plethora of available package managers. Also, as a result of the Mac OSX being an extension of the linux operating system and therefore provides all of linux's convenient programming utilities such as the command prompt, terminal commands and application interfacing using installation utilities such as pip and easy install.

`

6.3 Code Conventions

Coding conventions are a set of guidelines for a specific programming language that recommend programming style, practices and methods for each aspect of a piece program written in this language[36]. These conventions usually cover file organization, indentation, comments, declarations, statements, white space, naming conventions, programming practices, programming principles, programming rules of thumb, architectural best practices, etc. These are guidelines for software structural quality.

Software programmers are highly recommended to follow these guidelines to help improve the readability of their source code and make software maintenance easier. Conventions may be formalized in a documented set of rules that an entire team or company follows, or may be as informal as the habitual coding practices of an individual. Coding conventions are not enforced by compilers. So applicability of these conventions has no impact on the execution of the compiler.

6.3.1 Naming Convention

The names of the variables, methods, class have been chosen carefully so as to reflect the intended use of the variables and what they represent. These have been followed with the exception of loop counters which are used and discarded almost immediately. A name which consists of more than one word is written as a single component with first word in small letters and following words with first letter as a capital letter.

6.3.2 File Organization

This project uses a separate developer file for each module in use. All the files are stored in a common directory. Also, the resources used by the files which includes the data set for training the classifier, the MPEG-4 format traffic recording video upon which the project is performed and the intermediate data generated by the program itself such as storage files, video frames are all also stored in the same common directory.

`

Furthermore, the output that the program generates is stored in a separate path called as the project directory. All output files including key frames, violation instances captured and results extracted from the database are all stored in the project directory.

6.3.3 Properties Declarations

The properties are declared in standard python convention that is the keyword ‘def’ followed by the name of the property. All properties are so named to reflect the nature of the operation that they are required to perform in the system as a whole. This naming is applied to all the properties defined in the system with the exception of the starting point of execution or main function. Class properties are declared within the body of the class.

6.3.4 Class Declarations

The project follows an object oriented design. All database instances are simulated using class objects; these include simulating vehicles with all identifiable properties such as the name of the owner, the name of the vehicle etc. Listed below are the elements associated with defining a class in this system.

- **Class definition**: Description of common elements of every instance of a class.
- **Properties**: Data storage for class instances.
- **Methods**: Special functions that implement operations that are usually performed only on instances of the class.
- **Objects**: Instances of classes, which contain actual data values stored in the objects' properties.
- **Subclasses**: Classes that are derived from other classes and that inherit the methods, properties, and events from those classes.
- **Superclasses**: Classes that are used as a basis for the creation of more specifically defined classes

  .

6.3.5 Comments

Comments are necessary part of any coding conventions as it improves the understandability of the code developed. Comment lines begin with the character ‘#’, and anything after a

`

‘#’character is ignored by the python interpreter. The # character itself only tells the interpreter to ignore the remainder of the same line.

In the project files, commented areas are printed in green by default, so they should be easy to identify in the text wrangler editor application of the OSX platform. Comments for blocks of code are started by a ’’’ and are delimited by the same three single quotes.

Comments are useful for explaining what function a certain piece of code performs especially if the code relies on implicit or subtle assumptions or otherwise perform subtle actions. For example,

```
#function to simulate a one second delay
def delay():
    for i in range(0,10000000):
        i *= 1
```

6.4 Difficulties Encountered and Strategies Used to Tackle

The system carries out processing in two phases, pre and post processing. During the pre processing phase noise elimination and redundant frame removal is carried out. The removal of background noise and incorporation of non static objects in the background was a challenging task. Noise was eliminated using the clustering technique and the background is effectively modeled using a regulator value to vary the rate of averaging.

Also, during the post processing phase, which is the actual detection and identification phase, the standard cascade classifiers were producing inconsistent results [37]. Hence, we used a combination of Support Vector Machines and Histogram of Oriented Gradients technique to get more accurate results [38][39].

6.4.1 Management of the volumes that holds the image instances and data

The system generates many intermediate frames, images and data required for individual stages of processing during the execution of the program. These include generation of frames of the video for performing background subtraction, generation of non-redundant frames for recompilation into video, capturing instances of traffic violation from the recompiled video and finally intermediate data about vehicle objects performing the violation.

`

The original frames of the video are not written to disk and hence don't need to be managed, the non-redundant frames are stored in the project directory under the sub-directory full colored key frames, the instances of violation are also stored in the project directory under the sub-directory violation and the intermediate data is stored in files also contained in the project directory. The information about the vehicle objects is stored in a virtual python dictionary containing vehicle class instances indexed by their unique plate numbers.

6.4.2 Presenting of the system to the user

The system developed in this project is packaged and deployed into a stand alone application that requires standard python and openCV libraries to run. Also, to the user, the package has been presented in the form of a Graphical User Interface completely developed using openCV to avoid the need for support from extraneous third party applications. The main module is key board interactive and the sub modules once open in the Graphical User Interface, details very clearly, step by step, every action that the user needs to perform in order to obtain the desired output. The Graphical User Interface functions both in a modular fashion and as an integrated unit to enable the user to know and understand the output received from every phase of processing during the execution of the program.

6.4.3 Handling deprecated methods and new methods

As the program is developed keeping platform independence in mind, implementations of methods that perform similar functions from previous openCV libraries cannot be used. Such methods are called deprecated methods. These methods either must be overridden to be used for a user specific function or then newer standard openCV library implementations of the function needs to employed.

For future versions of python and openCV all the methods implemented in the system will either be available or deprecated. For deprecated methods, newer methods will be available as substitutions. But in the event that this does not happen, new methods have to defined and implemented, preferably added as a package to the standard openCV library for future use. In this system all methods that have been implemented from library files are defined in the python2.7 and above standard library and openCV 2.0 and above library.

`

6.5 Summary

This chapter presented the transformation of the design of the project into working model. The project was implemented on vim editor and text wrangler editor using the python programming language version 2.7 adhering to best coding policies and practices available. The implementation logic was devised formidably to tackle the impediments and hurdles that were encountered during the functioning of the violation detection system.

`

Chapter 7

Testing of object tracking for video analysis in traffic scenes

During and after the implementation of the system, the system must undergo continuous testing to ensure at every step that the system meets the requirements and needs of the user. After the implementation of the system is complete, it is necessary to test it in various scenarios to fix any bugs encountered during this process.

This process is called verification and validation of the system [40]. The primary purpose of testing is to identify bugs in the program and take corrective measures before it is ready for delivery as an application to the end user. It is an essential step in the process of finishing the system.

7.1 Testing Environment

Testing was done on a Macbook Pro 2014 Model, connected to a power supply with the following specifications:

Hardware Specifications:

- 2.2GHz quad-core Intel Core i7 processor (Turbo Boost up to 3.4GHz) with 6MB shared L3 cache12.00 GB RAM
- 64 bit OS
- 512 GB HardDisk

Software Specifications:

- Mac OSX Mavericks
- Vim editor, python2.7 and OpenCV 2.9

`

7.2 Unit Testing of Main Modules

Unit testing (sometimes called component testing) is the process of testing individual components in the system [41]. This is a defect testing process so its goal is to expose faults in these components. In system testing, unit testing is a method by which individual modules of the whole system, sets of one or more modules together with associated control data, usage procedures, and operating procedures, are tested to determine if they are fit for use. Intuitively, one can view a unit as the smallest testable part of an application or system. Unit tests are created by the developers of the system or occasionally by white box testers during the development process.

Unit Testing of Background Modeling Module

The Background Modeling unit is tested and test case is tabulated in the table. The test is performed to check whether the background modeling module is creating the scene background as specified by the rate of averaging of the video frames. The test was successful and the Table 7.1 references the tests:

Table 7.1: Unit Test Case 1 to check if Background Modeling succeeds

Sl No of Test Case	Traffic violation detection system – 1
Name of Test	Test to check if scene background is modeled convincingly.
Feature Being Tested	Create Scene Background.
Description	When the background modeling module is invoked, the video frames are averaged at a suitable rate to obtain the scene background.
Sample input	MPEG-4 format recording of vehicles jumping signal in a traffic scene.
Expected Output	Scene background with complete elimination of all foreground objects and incorporation of non-static background objects.
Actual Output	Scene background with complete elimination of all foreground objects and incorporation of non-static background objects.
Remarks	Test successfully completed.

`

The figure 7.1 shows the scene with the foreground object. Figure 7.2 shows the object being averaged out of the scene. Finally, figure 7.3 shows the modeled background without any moving objects.

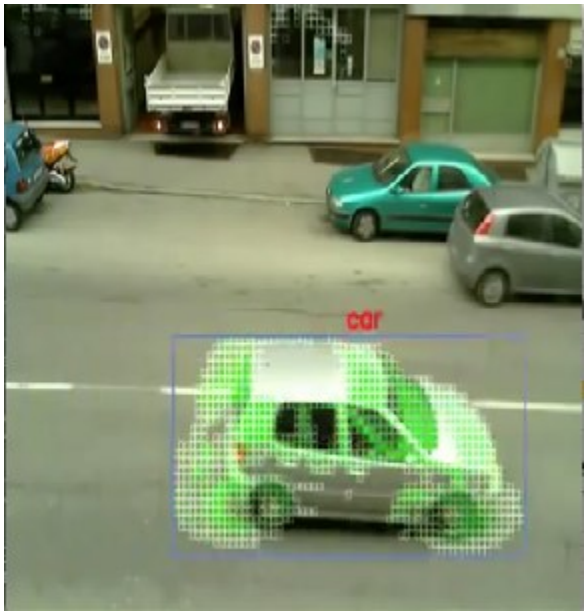

Figure 7.1: Scene with foreground object

Figure 7.2: Foreground object being removed

`

Figure 7.3: Modeled scene background

Unit Test of Foreground Modeling

The Foreground modeling unit is tested and test case is tabulated in the table. The test conducted was to check if the foreground objects are successfully modeled in the scene. The input to this module is the video and the background from the background modeling module. The test was successful. The test is referenced in table 7.2

Table 7.2: Unit Test Case 2 to check if the foreground modeling module succeeds

Sl No of Test Case	Traffic violation detection system-2
Name of Test	Test to check if Foreground modeling succeeds
Feature Being Tested	Extraction of foreground objects in the scene
Description	The video frames are subtracted against the modeled background to obtain the foreground.
Sample input	MPEG-4 format traffic video and 3 channel background image
Expected Output	MPEG-4 format video with modeled foreground
Actual Output	MPEG-4 format video with modeled foreground
Remarks	Test Successfully completed.

`

Figure 7.4 shows the original frame in the video and figure 7.5 shows the modeled foreground.

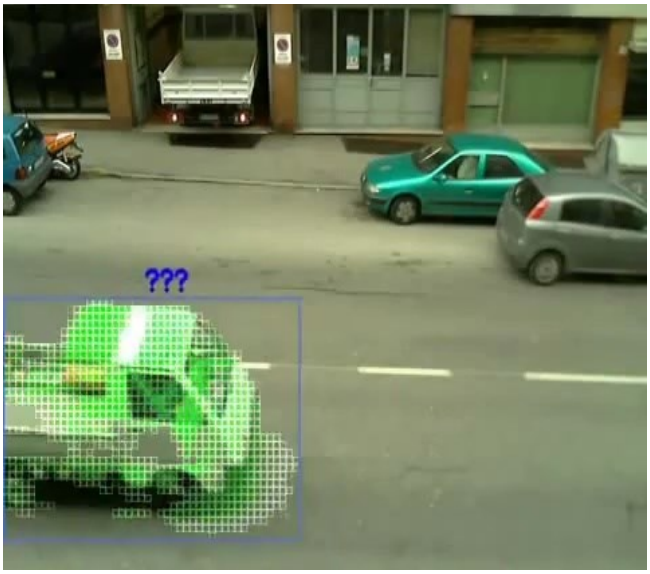

Figure 7.4: Original frame

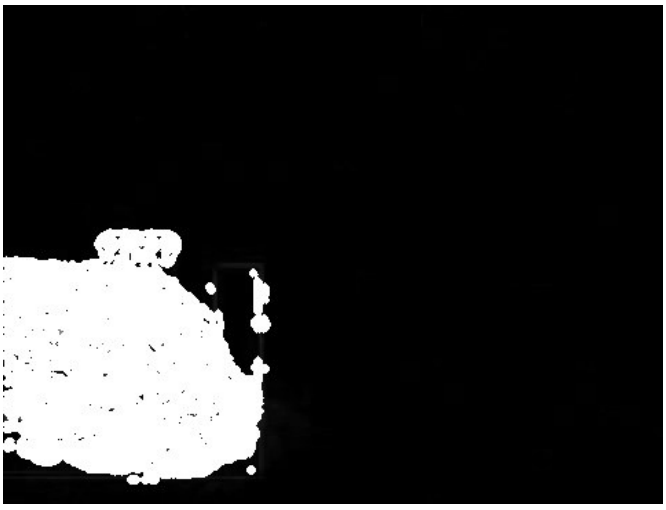

Figure 7.5: Frame with modeled foreground

Unit test of Color quantization and background noise elimination

In this unit test, the clustering based color quantization and background noise elimination module is tested and the test case is tabulated in the table. The test is takes input from the

`

foreground modeling phase and performs clustering on them. The results of the phase are supposed to make the foreground object clearer and eliminate background noise. The test is referenced in Table 7.3.

Table 7.3: Unit Test Case 3 check if Clustering and noise elimination succeeds

Sl No of Test Case	Traffic violation detection system-3.
Name of Test	Test to check if Clustering and noise elimination is achieved.
Feature Being Tested	Background noise elimination and foreground clarity.
Description	When the foreground frames are clustered the quantization of the image improves the clarity and eliminates all background noise. 2 clusters are used.
Sample input	MPEG-4 format video with modeled foreground with noise.
Expected Output	MPEG-4 format video with modeled foreground without noise and improved clarity.
Actual Output	MPEG-4 format video with modeled foreground without noise and improved clarity.
Remarks	Test completed successfully.

Figure 7.5 shows the foreground modeled and figure 7.6 show the results of clustering.

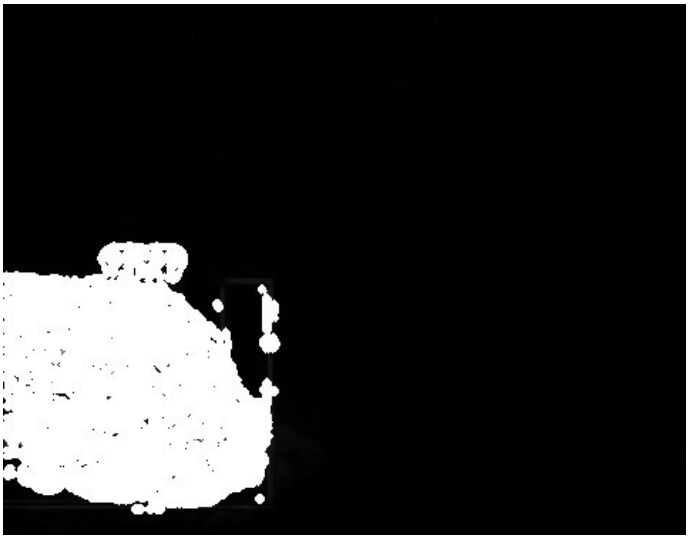

Figure 7.5: Foreground modeled

`

Figure 7.6: Detailed foreground frame with no noise

Unit testing the Keyframe extraction module

The Keyframe extraction module unit is tested and test case is tabulated in the table. The test conducted was to check whether redundant and blank frames are eliminated resulting in only frames that are useful in the next phase.. The test was successful. The test is referenced in table 7.4

Table 7.4: Unit Test Case 4 to check if key frames are extracted

Sl No of Test Case	Traffic violation detection system-4.
Name of Test	Test to check if key frames are extracted.
Feature Being Tested	Blank and redundant frame elimination.
Description	The module uses the results of the clustering and filters the result using a mean squared error similarity checker.
Sample input	MPEG-4 format foreground modeled video after clustering
Expected Output	MPEG-4 format foreground modeled video with elimination of blank and redundant frames
Actual Output	MPEG-4 format foreground modeled video with elimination of blank and redundant frames
Remarks	Test Successfully Completed.

`

Figure 7.7 shows the a few frames from the original foreground model with a blank frame between two frames of significant movement and figure 7.8 shows the result of key frame extraction.

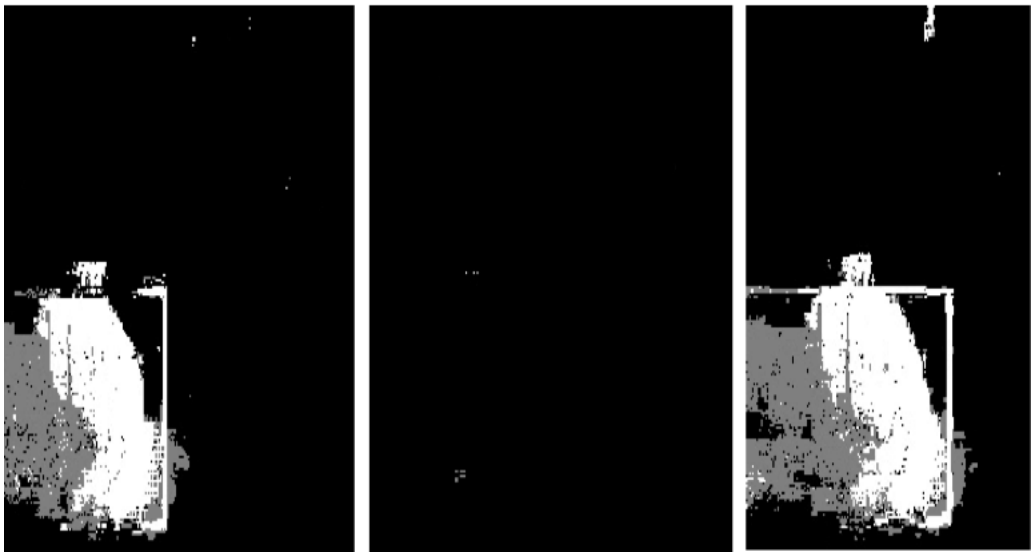

Figure 7.7: Frames with blank frame in between.

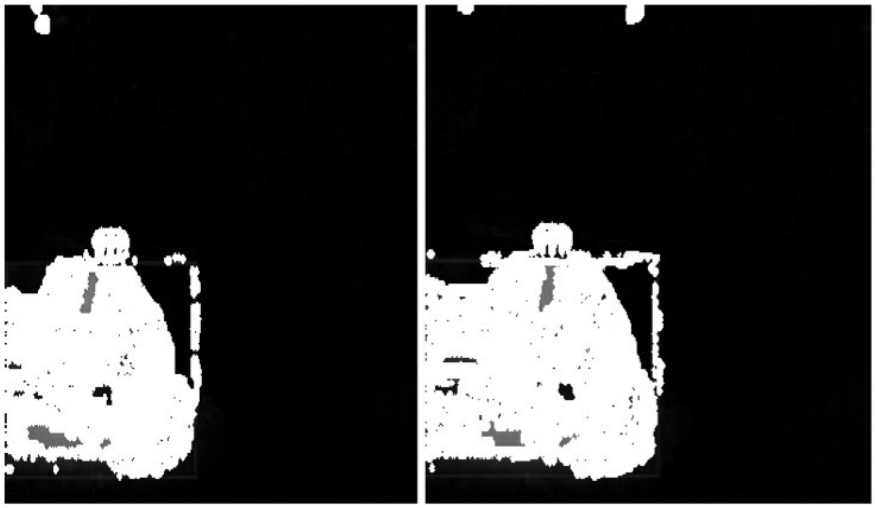

Figure 7.8: Key frames extracted.

`

Unit Testing of Violation detection Module.

The Violation detection Module is tested and test case is tabulated in the table. The test conducted was to detect the vehicle crossing the edge of the frame, capturing the frame instance and time stamping it. The test was successful. The test is referenced in table 7.5

Table 7.5: Unit Test Case 5 to check if violation detection is successful

Sl No of Test Case	Traffic violation detection system – 5.
Name of Test	Test to check if the violation is detected successfully.
Feature Being Tested	Capturing frame instance of video where violation occurs.
Description	When the vehicle in the frame crosses the edge of the frame, the frame is captured and time stamped.
Sample input	MPEG-4 format Recompiled video from the key frame extraction phase.
Expected Output	Captured frames showing the violation.
Actual Output	Captured frames showing the violation.
Remarks	Test Successfully Completed.

`

Figure 7.9 shows the instance of violation captured and time stamped. Figure 7.10 shows violation captured with two vehicle objects in the scene. Figure 7.11 shows the violation captured with a two wheeler vehicle.

Figure 7.9: Violation Captured

Figure 7.10: Violation captured with two vehicle objects.

`

Figure 7.11: Violation captured with a two wheeler vehicle

Unit Testing of plate number recognition of violating vehicle.

The Plate number recognition module is tested and test case is tabulated in the table. The test conducted was to check whether the number in the plate is recognized accurately by the digit classifier. The test was successful. The test is referenced in table 7.6

Table 7.6: Unit Test Case 6 to check if number recognition succeeds

Sl No of Test Case	Traffic Violation Detection System – 6
Name of Test	Test to check if the numbers on the plate are recognized
Feature Being Tested	Number recognition
Description	The numbers on the plate are recognized by a classifier. The classifier is trained initially using a digit data set and support vector machines.
Sample input	Plate extracted JPEG image
Expected Output	Number recognized as on plate
Actual Output	Number recognized as on plate
Remarks	Test Completed Successfully

`

Figure 7.12 and figure 7.13 shows the numbers being successfully recognized. Figure 7.14 shows the number being partially recognized. The program incorporates an intelligent string matching algorithm to obtain the desired database filtering.

Figure 7.12: Number recognized.

Figure 7.13: Number recognized.

Figure 7.14: Number partially recognized.

Unit Testing of filtering database results module

The Information filtering module is tested and test case is tabulated in the table. The test conducted was to check the number recognition results can successfully reduce the number of matched database results. The test was successful. The test is referenced in table 7.7

Table 7.7: Unit Test Case 7 to check database filtering

`

Sl No of Test Case	Traffic Violation Detection System – 7.
Name of Test	Test to check if the database results are successfully filtered.
Feature Being Tested	Vehicle match from database results.
Description	The database is indexed by the recognized numbers and any match found is displayed.
Sample input	Text file containing the number recognized.
Expected Output	JPEG image displaying vehicles matched and their details.
Actual Output	JPEG image displaying vehicles matched and their details.
Remarks	Test Successfully Completed.

Figure 7.15, figure 7.16 and figure 7.17 shows the results of database matching and vehicle details obtained. The vehicle information shown is extracted from the simulated vehicle database programmed manually in the system.

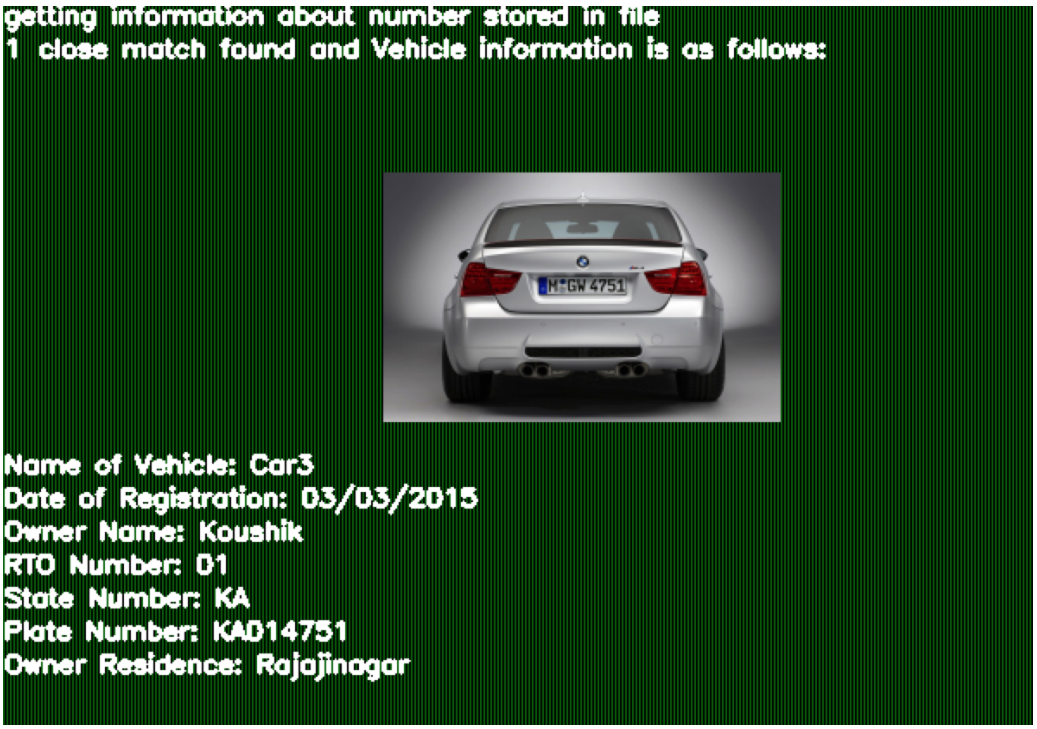

Figure 7.15: Database results.

`

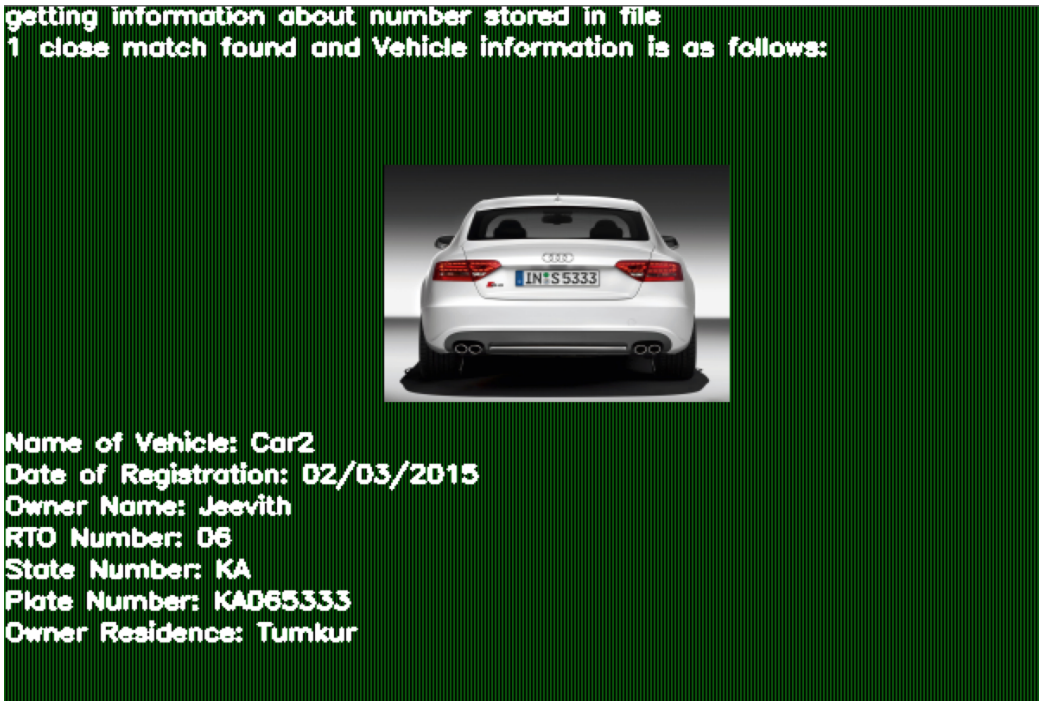

Figure 7.16: Database results.

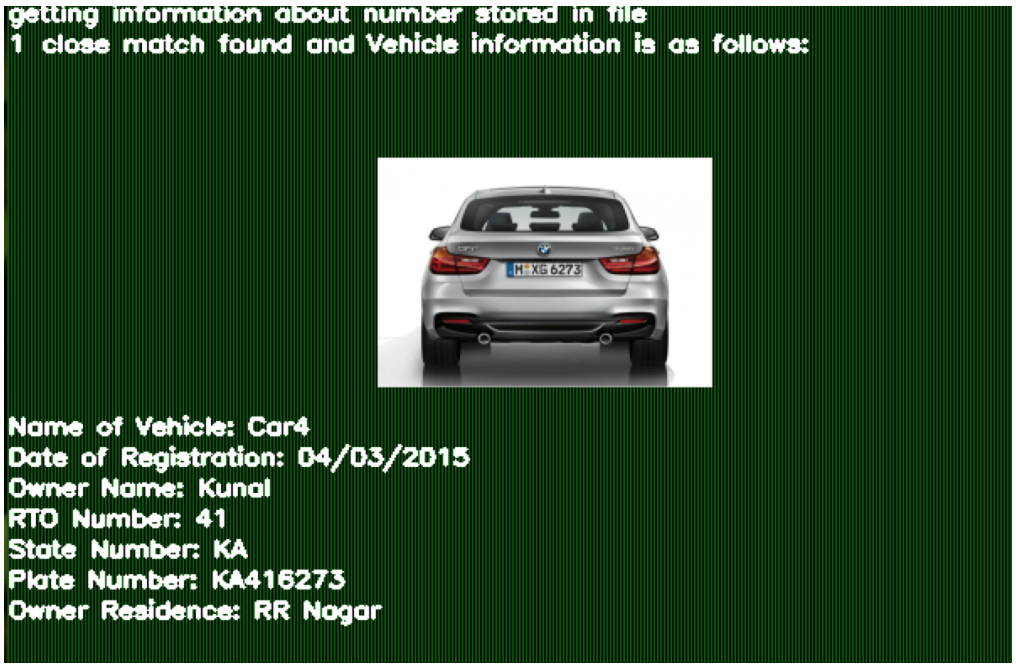

Figure 7.17: Database results.

`

7.3 Integration Testing Of Modules

The process of system integration involves building a system from its components and testing the resultant system for problems that arise from component interactions [42]. The components that are integrated may be off-the-shelf components, reusable components that have been adapted for a particular system or newly developed components. For many large systems, all three types of components are likely to be used. Integration testing checks that these components actually work together are called correctly and transfer the right data at the right time across their interfaces.

In integrated testing, the modules are joined to form sub-systems which each perform a specific function. The performance of the product is determined by its holistic performance. The data selected for testing needs to carefully selected [43]. The test was conducted to see if the violations in the video and appropriate database results are displayed. This operation is one which creates a frame to capture the violation according to the input parameters and presents the number plate to the digit classifier. Next the information is filtered from the database using the plate numbers. This test was found to be successful. This is referenced in table 7.8:

Table 7.8: Integrated Test Case 1 to Test case to see if Traffic violation detection system works.

Sl No of Test Case	Traffic Violation Detection System - 8
Name of Test	To check if the violation results are obtained from the input video
Feature Being Tested	The functioning of all unit modules to obtain violation results
Description	All the modules are run serially to obtain the violation results as output
Sample input	MPEG-4 format traffic scene recording
Expected Output	Violation details from the database
Actual Output	Violation details from the database
Remarks	Test Successfully completed

`

Integration testing of the system is carried out to check if the all the modules work together and communicate the right information between the interfaces connecting the modules. The test was successful and the system works as required.

7.4 System Testing

System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. System testing fails within the scope of black box testing, and as such, should require no knowledge of the inner design of the code or logic. The entire Traffic Violation Detection system is tested from start to finish, including testing of all conditions and permutations. The test was found to be successful. The Test Case is tabulated in Table 7.9.

Table 7.9: System testing of the Violation detection system

Sl No of Test Case	Traffic Violation Detection System -9.
Name of Test	System test.
Feature Being Tested	Traffic Violation Detection System.
Description	The flow of output between all the multiple stages of the system is tested with various types of input. The results obtained are checked for accuracy and consistency.
Sample input	Array details and input parameters to various operations.
Expected Output	All the required operations are performed successfully and the same is seen in the GUI of the vendors.
Actual Output	All the required operations are performed successfully and the same is seen in the GUI of the vendors.
Remarks	Test Successfully Completed.

The system test is used to ensure the compliance of all modules of the system. It tests to see if the various modules work in synchronization with each other to provide the expected results. If the test is a success, it indicates system is ready for use.

`

7.5 Summary

This chapter has described the various unit, integration and system tests conducted on the Traffic Violation Detection System. The input parameters, output results and description of each test is mentioned in the table representing the test. Finally, all tests have been passed and the system is ready for use.

`

Chapter 8

Experimental results and analysis of object tracking for video analysis in traffic scenes

The project was carried out on the Mac OSX operating system and has been compared to similar results obtained with the standard Gaussian Mixture Model algorithm. As the system is part of product development process, it was experimented for successful deployment. This chapter lists the result of the experiment conducted and the inference that were made from the testing. The evaluation metrics have been listed and the results have been accordingly quantified. The results that were got from the experiment describe the general performance trend of the Traffic Violation Detection System.

Evaluation Metric

The Traffic Violation Detection System was designed to propose a new method of detecting features from images that have enhanced clarity. Results of foreground Modeling obtained using the Gaussian Mixture Model have not been able to produce such clarity. Moreover, results of the Gaussian Mixture Model applied to variable frame rate videos have shown the insertion of blank and redundant frames. Therefore, the performance enhancement with respect to elimination of these blank and redundant frames has been chosen as an evaluation metric of this system. In addition to result comparison metrics with the Gaussian Mixture Model, the running of the clustering process in a serial and distributed fashion have also been compared, naturally using the time of execution of the processes as an evaluation metric.

Experimental Dataset

The dataset on which the system was tested is a MPEG-4 format video recording of a real life traffic scene. The system was also tested with the same converted to MOV and AVI formats. The image frames contained in the video all contain three channels, each represented either the Red, Green or Blue channels supported by the python implementation of OpenCV. The database information of vehicles recorded is stored in a simulated database modeled using the python

`

Object oriented model. The data set used for training the digit classifier was the MNIST handwritten digit database [49].

Performance Analysis

The results of obtaining foreground modeled frames from the recorded traffic scene are compared in the figure 8.1. This first result is the foreground modeled using the Gaussian Mixture Model and the second result is the foreground modeled using the proposed method. As is evident the number of pixels modeled in the foreground is higher in the second result as compared to the first. Also, the initial noise recorded is also much less in the second result as compared to the first result.

.

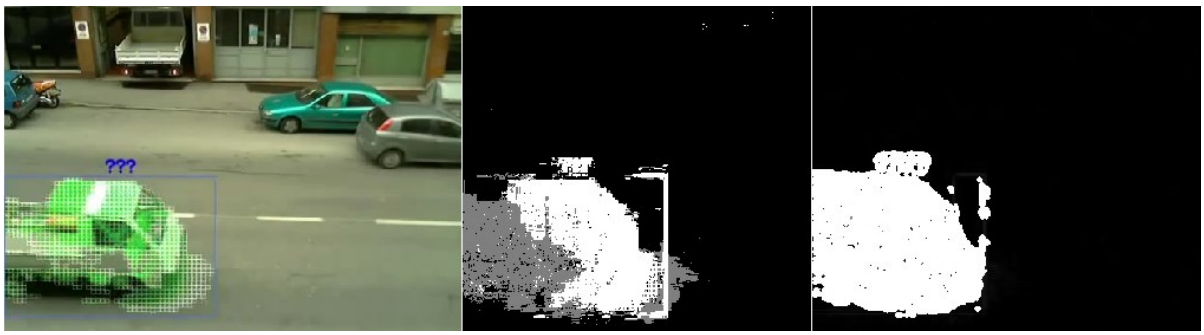

Figure 8.1: shows the original frame, the first result which is the result of GMM and second result which is the result of the proposed method.

Figure 8.2 shows the results after clustering the frames. The first result is that obtained by the GMM method, the second result is that obtained by clustering of the foreground frame. As can be seen, the second result very closely resembles the original frame in the amount of detail and clarity.

`

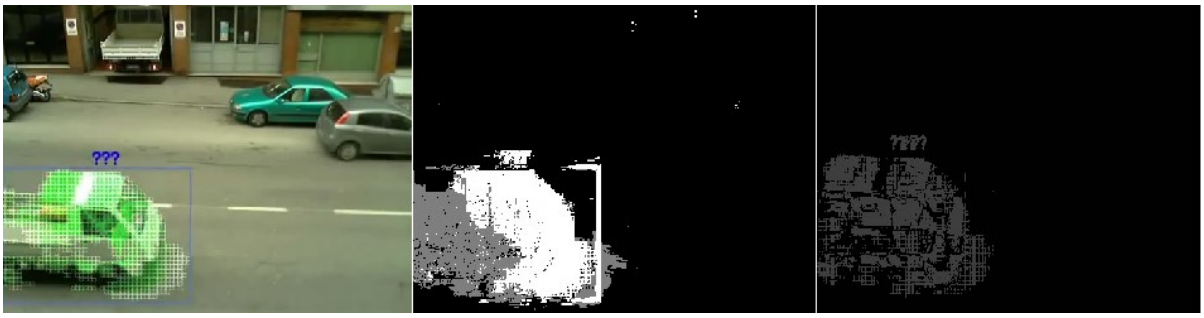

Figure 8.2: The figure shows the original frame, the foreground modeled by GMM and the foreground modeled using the proposed method and clustering.

Figure 8.3 shows the reduction in number of frames by the proposed method in comparision with that obtained using the Gaussian Mixture Model.

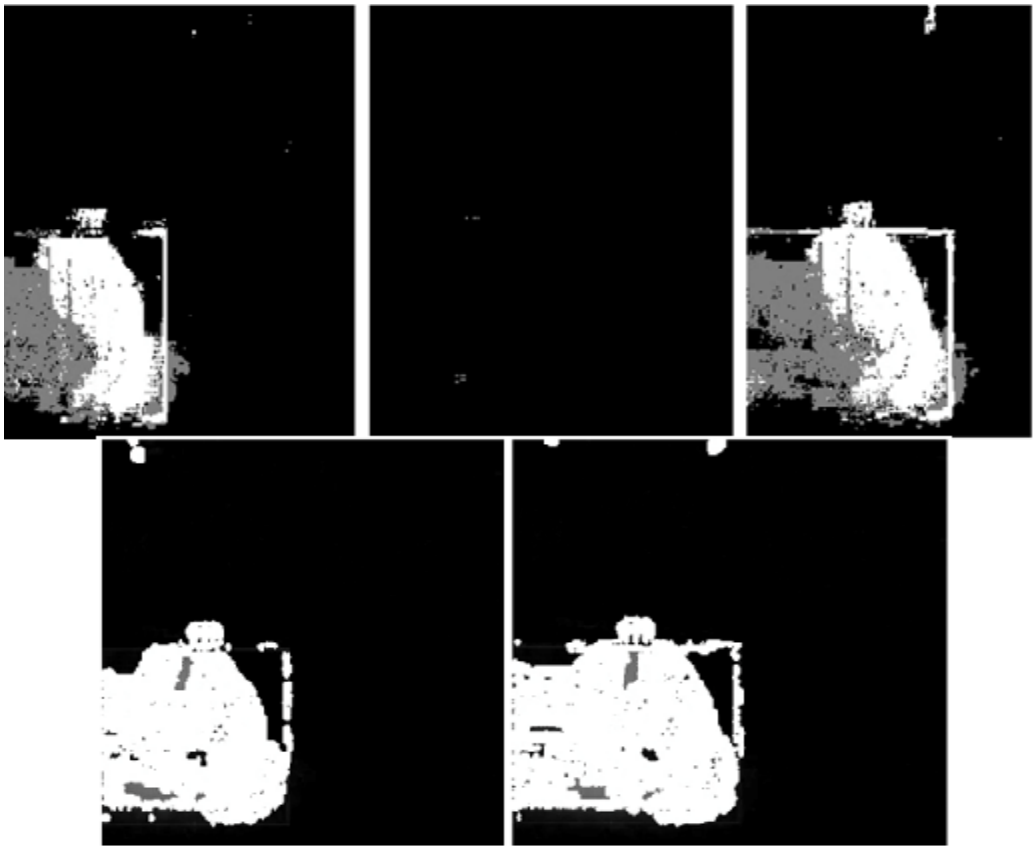

Figure 8.3: Shows the blank frame inserted using the GMM method, and the frame eliminated using the proposed method.

`

Inference from the Result

Program executed successfully under all testing scenarios. The accuracy of foreground modeling was 98.7% as measured by the percentage of moving pixels captured in the foreground that is number of moving pixels modeled divided by the total number of moving pixels divided by hundred. 100% noise elimination was achieved in the resulting foreground and the execution time increase by blank frame elimination was by a factor of approximately 4.2. This is calculated taking a sample of 42 frames. For 42 frames, 10 frames are extracted therefore; the improvement factor is 42/10 which is equal to 4.2. Also, the time required for the clustering process carried out in a distributed fashion was found to be increased by a factor of 2.5 as opposed to the same carried out using serial execution. This ratio was calculated by taking the time taken for clustering of a single frame in distributed fashion to the time taken for the same in serial fashion, which was found to be 2.5 seconds to 1. The plates and information gathered was accurate with respect to the simulated database. The percentage accuracy of recognizing the digits on the number plate was 88% calculated by taking 25 test cases (22 of them were successfully recognized). Therefore 22/25 = 0.88 or 88 percent accuracy.

Aspect of system considered	Result
Accuracy of Foreground Modeled	98.7 percentage
Noise elimination	100 percentage
Execution time improved (taken for 42 frames)	By a factor of 4.2
Execution time improved by parallelization	By a factor of 2.5
Accuracy of digits recognized (taken for 25 frames)	88 percentage

Summary

This chapter gave a perception on the performance of the project by analyzing the inference from the experimental result. The system was deployed across all platforms with the support libraries installed. The result of the system was spectacular with required operation being efficaciously accomplished and the results showing a marked improvement over already existing models.

`

Chapter 9

Conclusion

The proposed system makes use of the K-Means Clustering algorithm and the Mean Square Error Comparison algorithm to achieve enhancement of the quality of the recorded traffic footage, specifically the elimination of background noise, blank frames and redundant frames. Also, it incorporates the efficient use of Support Vector Machines and Histogram of Oriented Gradients to extract meaningful database information from the captured instance of vehicle violation. The system has been successfully tested in various traffic scenarios and the results have been shown to better than similar results obtained using the Gaussian Mixture Model, which is the de-facto standard of object tracking in videos in the Image processing community.

Limitations

The Traffic Violation Detection System does suffer from a few limitations.

- It requires that the rear view of the vehicle responsible for violation, be captured by a secondary camera of reasonably high quality.
- Moreover, the method requires that the scene be captured from a lateral view, as it has been found suitable for application of the proposed system with maximum computational gain and simplicity.
- Finally, the system requires the original quality of the recorded video to be of decent clarity to ensure accurate results, even though it includes a set of modules dedicated to enhancing the quality of the recorded video.

Future Enhancement

- Ongoing research aimed at developing the system further to reduce its limitations deals with trying to reduce the role of the secondary camera required for capturing the rear view of the vehicle performing the signal jump.

`

- This can be accomplished to a certain extent using the method triangle similarity in which the distance from the camera to a marker on the object of interest can be calculated using physical optics concepts of focal length.
- Also, to further increase the accuracy of the vehicle identification results filtered from the simulated database, an alphanumeric dataset can be compiled to train a classifier that can recognize all the characters contained in a typical number plate. However, identification of local histogram patterns for such a complex image presents a challenging problem.