VISVESVARAYA TECHNOLOGICAL UNIVERSITY
"JNANA SANGAMA", BELAGAVI-590018.

A
Project Report
on

# "NEURAL NETWORK BASED VEHICLE CLASSIFICATION FOR INTELLIGENT TRAFFIC CONTROL"

Submitted in partial fulfillment for the award of the degree of
BACHELOR OF ENGINEERING
IN
COMPUTER SCIENCE AND ENGINEERING

By

| | |
|---|---|
| SACHIN POKHARIA | [1JB13CS130] |
| SMITHA K | [1JB13CS155] |
| VAISHNAVI R HEGDE | [1JB13CS169] |
| VARUN KUMAR H N | [1JB13CS170] |

UNDER THE GUIDANCE OF

Mrs. Divyashree J
Assistant Professor
Department of CSE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SJB INSTITUTE OF TECHNOLOGY
No. 67, BGS Health & Education City, Dr. Vishnuvardhan Road
Kengeri, Bengaluru- 560060
2016 - 2017

||Jai Sri Gurudev||

Sri Adichunchanagiri Shikshana Trust ®

**SJB INSTITUTE OF TECHNOLOGY**

**No. 67, BGS Health & Education City, Dr. Vishnuvardhan Road**
**Kengeri, Bengaluru- 560060**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

# CERTIFICATE

Certified that the project work entitled **"NEURAL NETWORK BASED VEHICLE CLASSIFICATION FOR INTELLIGENT TRAFFIC CONTROL"** carried out by **SACHIN POKHARIA [1JB13CS130], SMITHA K [1JB13CS155], VAISHNAVI R HEGDE [1JB13CS169], VARUN KUMAR H N[1JB13CS170]** are bonafide students of **SJB Institute of Technology** in partial fulfillment for the award of **"BACHELOR OF ENGINEERING"** in **Computer Science and Engineering** as prescribed by **VISVESVARAYA TECHNOLOGICAL UNIVERSITY, BELAGAVI** during the academic year **2016-2017**. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the Report deposited in the Departmental library. The project report has been approved as it satisfies the academic requirements in respect of Project work prescribed for the said Degree.

| | | |
|---|---|---|
| **Signature of Guide** | **Signature of HOD** | **Signature of Principal** |
| **Mrs. Divyashree J** | **Dr. Krishna A N** | **Dr. PUTTARAJU** |
| **Assistant Professor** | **Professor & Head** | **Principal** |
| **Dept. of CSE** | **Dept. of CSE** | **SJBIT, Bangalore** |

**EXTERNAL VIVA**

| | |
|---|---|
| **Name of the examiners** | **Signature with date** |
| **1.** ____________________ | ____________________ |
| **2.** ____________________ | ____________________ |

# ACKNOWLEDGEMENT

We would like to express our profound grateful to His Divine Soul **Jagadguru Padmabhushan Sri Sri Sri Dr. Balagangadharanatha Mahaswamiji** and His Holiness **Jagadguru Sri Sri Sri Dr. Nirmalanandanatha MahaSwamiji** for providing us an opportunity to complete our academics in this esteemed institution.

We would also like to express our profound thanks to **Reverend Sri Sri Prakashnath Swamiji**, Managing Director, SJB Institute of Technology, for his continuous support in providing amenities to carry out this project in this admired institution.

We express our gratitude to **Dr. Puttaraju,** Principal, SJB Institute of Technology, for providing us an excellent facilities and academic ambience; which have helped us in satisfactory completion of project work.

We extend our sincere thanks to **Dr. Krishan A N,** Head of the Department, Computer Science and Engineering for providing us an invaluable support throughout the period of our project work.

We wish to express our heartfelt gratitude to our guide **Mrs. Divyashree J,** for her valuable guidance, suggestions and cheerful encouragement during the entire period of our project work.

We express our truthful thanks to **Dr. Naveena C, Project Coordinator**, Department of CSE, for his valuable support.

Finally, we take this opportunity to extend our earnest gratitude and respect to our parents, Teaching & Non teaching staffs of the department, the library staff and all our friends, who have directly or indirectly supported us during the period of our project work.

Regards,

| | |
|---|---|
| SACHIN POKHARIA | [1JB13CS130] |
| SMITHA K | [1JB13CS155] |
| VAISHNAVI R HEGDE | [1JB13CS169] |
| VARUN KUMAR H N | [1JB13CS170] |

# ABSTRACT

This project presents a novel method of vehicle classification using parameterized model and neural networks. A parameterized model is presented, which can describe features of vehicle. In this model, vertices and their topological structure are regarded as the key features. Then a classifier based on multi-layer perception networks (MLPN) is adopted to recognize vehicles.

In this project, an immovable camera has been used which is located in nearly close height of the road surface to detect and classify the vehicles. The algorithm that used is included two general phases; at first, mobile vehicles in the traffic situations are obtained by using some techniques that include image processing and remove background of the images and performing edge detection and morphology operations. In the second phase, vehicles near the camera are selected and the specific features are processed and extracted. These features apply to the neural networks as a vector so the outputs determine type of vehicle. This presented model is able to classify the vehicles in three classes; heavy vehicles, light vehicles and motorcycles. Results demonstrate accuracy of the algorithm and its highly functional level.

Experimental results show that parameterized model can satisfactorily and effectively describe vehicles. This novel method can be used in real world systems such as the vehicle verifying system in toll collecting station. However, it is not difficult to adapt algorithms and improve model to fit for other traffic scene.

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# INTRODUCTION

Chapter 1

# INTRODUCTION

## 1.1 IMAGE PROCESSING

Image processing is a method to convert an image into digital form and perform some operations on it, in order to get an enhanced image or to extract some useful information from it. It is a type of signal dispensation in which input is image, like video frame or photograph and output may be image or characteristics associated with that image. Usually Image Processing system includes treating images as two dimensional signals while applying already set signal processing methods to them.

It is among rapidly growing technologies today, with its applications in various aspects of a business. Image Processing forms core research area within engineering and computer science disciplines too.

Image processing basically includes the following three steps.

- Importing the image with optical scanner or by digital photography.
- Analyzing and manipulating the image which includes data compression and image enhancement and spotting patterns that are not to human eyes like satellite photographs.
- Output is the last stage in which result can be altered image or report that is based on image analysis.

### 1.1.1 PURPOSE OF IMAGE PROCESSING

The purpose of image processing is divided into 5 groups. They are:

1. Visualization - Observe the objects that are not visible.
2. Image sharpening and restoration - To create a better image.
3. Image retrieval - Seek for the image of interest.
4. Measurement of pattern – Measures various objects in an image.
5. Image Recognition – Distinguish the objects in an image.

### 1.1.2 TYPES

The two types of methods used for Image Processing are **Analog** and **Digital Image Processing**. Analog or visual techniques of image processing can be used for the hard copies like printouts and photographs. Image analysts use various fundamentals of interpretation while using these visual techniques. The image processing is not just confined to area that has to be studied but on knowledge of analyst. Association is another important tool in image processing through visual techniques. So analysts apply a combination of personal knowledge and collateral data to image processing.

Digital Processing techniques help in manipulation of the digital images by using computers. As raw data from imaging sensors from satellite platform contains deficiencies. To get over such flaws and to get originality of information, it has to undergo various phases of processing. The three general phases that all types of data have to undergo while using digital technique are Pre- processing, enhancement and display, information extraction**.**

## 1.2 NEURAL NETWORKS

Artificial neural networks (ANNs) or connectionist systems are a computational model used in machine learning, computer science and other research disciplines, which is based on a large collection of connected simple units called artificial neurons, loosely analogous to axons in a biological brain. Connections between neurons carry an activation signal of varying strength. If the combined incoming signals are strong enough, the neuron becomes activated and the signal travels to other neurons connected to it. Such systems can be trained from examples, rather than explicitly programmed, and excel in areas where the solution or feature detection is difficult to express in a traditional computer program. Like other machine learning methods, neural networks have been used to solve a wide variety of tasks, like computer vision and speech recognition, that are difficult to solve using ordinary rule-based programming.

Typically, neurons are connected in layers, and signals travel from the first (input), to the last (output) layer. Modern neural network projects typically have a few thousand to a few million neural units and millions of connections; their computing power is similar to a worm brain, several orders of magnitude simpler than a human brain. The signals and state of artificial neurons are real numbers, typically between 0 and 1. There may be a threshold function or limiting function on each connection and on the unit itself, such that the signal must surpass the limit before propagating. Back propagation is the use of forward stimulation to modify connection weights, and is sometimes done to train the network using known correct outputs. However, the success is unpredictable: after training, some systems are good at solving problems while others are not. Training typically requires several thousand cycles of interaction.

The goal of the neural network is to solve problems in the same way that a human would, although several neural network categories are more abstract.

## 1.3 OUR APPROACH

Recently, researchers pay more attention to the technologies of vehicle recognition. Vehicle recognition plays an important role in the fields of road traffic monitoring and management. For example, in the toll collecting system on highway, vehicles should be classified and verified, because the toll collecting depends on the type of vehicles.

Traffic management and information systems rely on a suite of sensors for estimating traffic parameters. Currently, magnetic loop detectors are often used to count vehicles passing overthem. Vision basedvideo monitoringsystemsoffer a number of advantages. In addition to vehicle counts, a much larger set of traffic parameters such as vehicle classifications, lane changes, etc., can be measured. Besides, cameras are much less disruptive to install than loop detectors. Vehicle classification is important in the computation of the percentages of vehicle classes that use state-aid streets and highways. The current situation is described by outdated data and often, human operators manually count vehicles at a specific street. The use of an automated system can lead to accurate design of pavements (e.g., the decision about thickness) with obvious results in cost and quality. Even in large metropolitan areas, there is a need for data about vehicle classes that use a particular street.

Vision-basedvideo monitoringsystemsoffer a number of advantages. In addition to vehicle counts, a much larger set of traffic parameters such as vehicle classifications, lane changes, etc., can be measured. Vehicle classification is important in the computation of the percentages of vehicle classes that use state-aid streets and highways. The use of an automated system can lead to accurate design of pavements (e.g., the decision about thickness) with obvious results in cost and quality. Even in large metropolitan areas, there is a need for data about vehicle classes that use a particular street. A classification system like the one proposed here can provide important data for a particular design scenario.

The typical method for vehicle recognition is based on modeling technologies. 3D structured models often are used to describe the vehicle shapes and structures. In these 3D models, point, lines, faces and topological structures are regarded as the key features to deal with the problems of translating, rotating, scaling and occluding. In the case of road traffic, vehicles are normally confined to be upright in contact with the road, and this ground-plane constraint (GPC) can reduce the pose recovery problem from 6 to 3 degree of freedom, which are translation (X,Y) on the ground-plane and rotation angle about the vertical axis.

In these methods, the vehicles are recognized by searching the pose space and by matching the features of model and the features in image. Unfortunately, the computational cost for pose space searching is expressive. This system uses a single camera mounted on a pole or other tall structure, looking down on the traffic scene. It can be used for detecting and classifying vehicles in multiple lanes and for any direction of traffic flow. The system requires only the camera calibration parameters and direction of traffic for initialization.

Based on the 3D structured model, this project first proposes a method based on parameterized model, which can represent the features of vehicles, and while gives an improved method to reduce the computational cost. Then we present a method based on the neural network to classify vehicles. Finally, the experimental results are presented which show that our method can effectively deal with vehicle recognition, and are satisfy to the requirement of real world system.

# LITERATURE SURVEY

Chapter 2

# LITERATURE SURVEY

Applying image processing technologies to vehicle detection has been a hot focus of research in Intelligent Transportation Systems (ITS) over the last decade. Tracking moving vehicles in video streams has been an active area of research in computer vision.

In [2], a real time system for measuring traffic parameters is described. It uses a feature-based method along with occlusion reasoning for tracking vehicles in congested traffic scenes. In order to handle occlusions, instead of tracking entire vehicles, vehicle sub-features are tracked. This approach however is very computationally expensive.

In [6], a moving object recognition method is described that uses an adaptive background subtraction technique to separate vehicles from the background. The background is modeled as a slow time-varying image sequence, which allows it to adapt to changes in lighting and weather conditions.

In a related work described in [10], pedestrians are tracked and counted using a single camera. The images from the input image sequence are segmented using background subtraction. The resulting connected regions are then grouped together into pedestrians and tracked. Merging and splitting of regions is treated as a graph optimization problem.

In [11], a system for detecting lane changes of vehicles in a traffic scene is introduced. The approach is similar to the one described in [10] with the addition that trajectories of the vehicles are determined to detect lane changes. Despite the large amount of literature on vehicle detection and tracking, there has been relatively little work done in the field of vehicle classification. This is because vehicle classification is an inherently hard problem. Moreover, detection and tracking are simply preliminary steps in the task of vehicle classification. Given the wide variety of shapes and sizes of vehicles within a single category alone, it is difficult to categorize vehicles using simple parameters. This task is made even more difficult when multiple categories are desired. In real-world traffic scenes, occlusions, shadows, camera noise, changes in lighting and weather conditions, etc. are a fact of life. In addition, stereo cameras are rarely used for traffic monitoring.

This makes the recovery of vehicle parameters—such as length, width, height etc., even more difficult given a single camera view. The inherent complexity of stereo algorithms and the need to solve the correspondence problem makes them unfeasible for real-time applications.

In [9], a vehicle tracking and classification system is described that can categorize moving objects as vehicles or humans. However, it does not further classify the vehicles into various classes.

In [7], an object classification approach that uses parameterized 3-D models is described. The system uses a3-D polyhedral model to classify vehicles in a traffic sequence. The system uses a generic vehicle model based on the shape of a typical sedan. The underlying assumption being that in typical traffic scenes, cars are more common than trucks or other types of vehicles. The University of Reading has done extensive work in three-dimensional tracking of vehicles and classification of the tracked vehicles using 3-D model matching methods.

Baker and Sullivan [1] and Sullivan [13] utilized knowledge of the camera calibration and that vehicles move on a plane in their 3-D model-based tracking. Three-dimensional wireframe models of various types of vehicles (e.g., sedans, hatchbacks, wagons, etc.) were developed. Projections of these models were then compared to features in the image.

In [15], this approach was extended so that the image features act as forces on the model. This reduced the number of iterations and improved performance. They also used parameterized models as deformable templates and used principal component analysis to reduce the number of parameters. Sullivan et al. [14] developed a simplified version of the same approach using orthographic approximations to attain real-time performance.

Lai et al. demonstrated that accurate vehicle dimension estimation could be performed through the use of a set of coordinate mapping functions. This method requires camera calibration in order to map image angles and pixels into real-world dimensions. Similarly, commercially available Video Image Processors (VIPs), such as the VideoTrack system developed by Peek Traffic Inc., are capable of truck data collection. However, the cost for such systems is significant and they require calibrated camera images to work correctly.

Calibrating these systems normally requires very specific road surface information and camera information which may not be easy to obtain.

Gupte et al. [3] performed similar work by instead tracking regions and using the fact that all motion occurs in the ground plane to detect, track, and classify vehicles. Hasegawa and Kanade [4] developed a system capable of detecting and classifying moving objects by both type and color. Vehicles from a series of training images were identified by an operator to develop the characteristics associated with each object type. In a test of 180 presented objects, 91% were correctly identified. A major disadvantage of this system, however, is the requirement for training images from the location of interest.

The project also involved various challenging concepts such as image processing, which, individually are vast areas of research. Some of the works that provide a sound understanding are:

Background subtraction is a widely used approach for detecting moving objects from static cameras. Many different methods have been proposed over the recent years and both the novice and the experts can be confused about their benefits and limitations. In order to overcome this problem, **Massimo Piccardi's** work provides a review of the main methods and an original categorization based on speed, memory requirements and accuracy, such a review can effectively guide the designer to select the most suitable method for a given application in a principled way. Methods reviewed include parametric and non-parametric background density estimates and spatial correlation approaches.

There are many challenges in developing a good background subtraction algorithm. First, it must be robust against changes in illumination. Second, it should avoid detecting non-stationary background objects such as swinging leaves, rain, snow, and shadow cast by moving objects. Finally, its internal background model should react quickly to changes in background such as starting and stopping of vehicles. **SenChing et al**. compare various background subtraction algorithms for detecting moving vehicles and pedestrians in urban traffic video sequences. They consider approaches varying from simple techniques such as frame differencing and adaptive median filtering, to more sophisticated probabilistic modeling techniques. While complicated techniques often produce superior performance, these experiments show that simple techniques such as adaptive median filtering can produce good results with much lower computational complexity.

**ZoranZivkovic et al.** analyze the usual pixel-level approach. They have developed an efficient adaptive algorithm using Gaussian mixture probability density. Recursive equations are used to constantly update the parameters and but also to simultaneously select the appropriate number of components for each pixel.

**T. Horprasert et al**. present a novel algorithm for detecting moving objects from a static background scene that contains shading and shadows using color images. They have developed a robust and efficiently computed background subtraction algorithm that is able to cope with local illumination changes, such as shadows and highlights, as well as global illumination changes. The algorithm is based on a proposed computational color model which separates the brightness from the chromaticity component. They have applied this method to real image sequences of both indoor and outdoor scenes. The results, which demonstrate the system's performance, and some speed up techniques they have employed in their implementation are also shown.

**Oliver et al.** present a technique for motion detection that incorporates several innovative mechanisms. Their proposed technique stores, for each pixel, a set of values taken in the past at the same location or in the neighborhood. It then compares this set to the current pixel value in order to determine whether that pixel belongs to the background, and adapts the model by choosing randomly which values to substitute from the background model. This approach differs from those based on the classical belief that the oldest values should be replaced first. Finally, when the pixel is found to be part of the background, its value is propagated into the background model of a neighboring pixel. They describe their method in full details (including the parameter values used) and compare it to other background subtraction techniques. Efficiency figures show that their method outperforms recent and proven state-of-the-art methods in terms of both computation speed and detection rate. They also analyze the performance of a downscaled version of their algorithm to the absolute minimum of one comparison and one byte of memory per pixel. It appears that even such a simplified version of their algorithm performs better than mainstream techniques. An implementation of ViBe is available at http://www.motiondetection.org

**Xue Mei and Haibin Ling** propose a robust visual tracking method by casting tracking as a sparse approximation problem in a particle filter framework. In this framework, occlusion, noise and other challenging issues are addressed seamlessly through a set of trivial templates. Specifically, to find the tracking target in a new frame, each target candidate is sparsely represented in the space spanned by target templates and trivial templates. The sparsity is achieved by solving an `1-regularized least squares problem. Then the candidate with the smallest projection error is taken as the tracking target. After that, tracking is continued using a Bayesian state inference framework.

Two strategies are used to further improve the tracking performance. First, target templates are dynamically updated to capture appearance changes. Second, non-negativity constraints are enforced to filter out clutters which negatively resemble tracking targets. They test the proposed approach on numerous sequences involving different types of challenges including occlusion and variations in illumination, scale, and pose. The proposed approach demonstrates excellent performance in comparison with previously proposed trackers. They also extend the method for simultaneous tracking and recognition by introducing a static template set, which stores target images from different classes. The recognition result at each frame is propagated to produce the final result for the whole video. The approach is validated on a vehicle tracking and classification task using outdoor infrared video sequences.

While numerous approaches have been introduced for this purpose, no specific study has been conducted to provide a robust and complete video-based vehicle classification system based on the rear-side view where the camera's field of view is directly behind the vehicle. **MehranKafai and BirBhanu** present a stochastic multiclass vehicle classification system which classifies a vehicle (given its direct rear-side view) into one of four classes: sedan, pickup truck, SUV/minivan, and unknown. A feature set of tail light and vehicle dimensions is extracted which feeds a feature selection algorithm to define a low-dimensional feature vector. The feature vector is then processed by a hybrid dynamic Bayesian network to classify each vehicle.

Although several commercial video image processing systems have been developed for traffic data collection, these systems are typically subject to several major problems including complicated calibration processes, poor detection accuracy under certain weather and lighting conditions, etc. Nonetheless, these previous investigations provide valuable insights to the video based vehicle detection and classification problems to be addressed in this study. This project aims at developing a new video-based vehicle detection and classification system for convenient and reliable traffic data collection using images captured by the non-calibrated video cameras.

# REQUIREMENT SPECIFICATION

Chapter 3

# REQUIREMENT SPECIFICATION

The requirement specification can be categorized and viewed from two different perspectives, they are:

- **Functional requirements**

  These are the statements of services the system should provide, how the system should react to particular inputs and how the system should behave in a particular situation. These depend on the type of software being developed, the expected users of the software. They may also reflect user's requirements.

- **Non Functional requirements**

  As the name suggests these requirements are those which are not directly concerned with the specific functions delivered by the system. These are constraints on the services or functions offered by the system. They include timing constraints, constraints on the development process, standards, response time, reliability and storage.

## 3.1 Functional Requirements

- The system should be able to detect vehicles in a video stream.
- Isolate the object of interest.
- Classify object of interest based on the knowledge gained by continuous training.
- To find the number and the type of vehicles.

## 3.2 Non Functional Requirements

**Usability**

Simplicity is the key here. The system must be simple such that people like to use it and not so complex that people avoid using it. The user must be familiar with the user interfaces and should not have problems in migrating to a new system with a new environment. The menus, buttons and dialog boxes should be named in a manner that they provide clear understanding of the functionality.

**Reliability**

The system should be trustworthy and reliable in providing the functionalities. The changes made by the programmer should be visible both to the project leader as well as the test engineer.

**Security**

Apart from bug tracking the system must provide necessary security and must secure the whole process from crashing. As technology began to grow in fast rate the security became the major concern of an organization. Millions of dollars are invested in providing security. Bug tracking delivers the maximum security available at the highest performance rate possible, ensuring that unauthorized users cannot access vital issue information without permission. Bug tracking system issues different authenticated users their secret passwords so that there are restricted functionalities for all the users.

**Scalability**

The system should be scalable enough to add new functionalities at a later stage. There should be a common channel, which can accommodate the new functionalities.

**Maintainability**

The system monitoring and maintenance should be simple and objective in its approach. There should not be too many jobs running on different machines such that it gets difficult to monitor whether the jobs are running without errors.

**Portability**

The system should be easily portable to another system. This is required when the web server, which is hosting the system gets stuck due to some problems, which requires the system to be taken to another system.

**Flexibility**

The system should be flexible enough to allow modifications at any point of time.

**Reusability**

The system should be divided into such modules that it could be used as a part of another system without requiring much of work.

Some of the non-functional requirements for our project include:

- Accurate classification
- Modifiable and adaptable to other traffic situations
- Minimal development cost as it has to be implemented on a large scale.

## 3.3 Software Requirements

The software requirements are as specified in the following table:

| | |
|---|---|
| Operating System | Windows 7 and above |
| Front End | MATLAB |
| Back End | MATLAB |
| Recommended | Windows 8 or higher |

**Table 3.1 Software Requirements**

## 3.4 Hardware Requirements

The hardware requirements are as specified in the following table:

| | |
|---|---|
| CPU Type and Speed | Core i3 or higher, 1.5 GHz or faster |
| Memory | 4 GB RAM |
| Hard Disk | About 5GB of free space |

**Table 3.2 Hardware Requirements**

## 3.5 Working environment requirement

MATLAB (matrix laboratory) is a multi-paradigm numerical computing environment and fourth-generation programming language. A proprietary programming language developed by MathWorks, MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages, including C, C++, C#, Java, Fortran and Python.

Since we are trying to extract features of the vehicles as individual data elements, they are conveniently placed in matrices. As MATLAB considers every data element as a matrix, it is extremely well equipped to handle matrix operations. We also make use of MATLAB to create and train the neural network. The major advantage lies in MATLAB's ability to generate a comprehensive set of plots to describe the output of the modules.

# SYSTEM ARCHITECTURE

# Chapter 4

# SYSTEM ARCHITECTURE

## 4.1 PROBLEM FORMULATION

More than 60000 vehicles are registered every day in India. With this number set to only increase further in coming years, effective traffic management is the concern of the hour. A move towards automation may help in reducing the burden to a greater extent. Newer avenues of technology like neural networks offer promising solutions to these problems. Although computers are good at learning rote things like maintaining databases, the ability to recognise and learn a pattern provides a very interesting dimension to this field.

This project aims at developing an intelligent system to decongest traffic at toll booths, by implementing a method that can detect vehicles from a video stream and classify the same into the said categories, thereby eliminating the need for human intervention.

## 4.2 PROPOSED ARCHITECTURE

Figure 4.1 illustrates the architecture that has been proposed to tackle the problem that has been identified above.

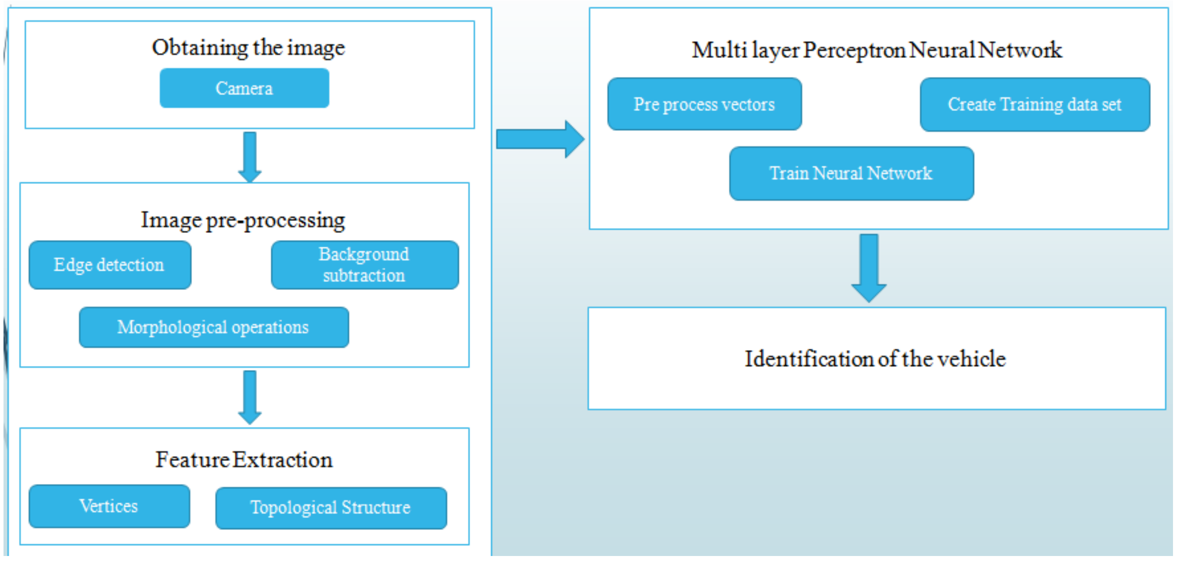


**Fig 4.1:** Framework for detection and classification

### 4.2.1 Obtaining the image

The major step of this model is data collection, and that is done by using a stationary camera placed at anoptimal height to capture the videos of vehicles passing by. For thisproject, we have only considered a stationary camera as we aim to implement the model to assist the detection and classification of vehicles at tool booths. Also, the involvement of a non-stationary camera leads to several complications. The camera is placed such that it can capture vehicles at a distance of 60-100 meters. This helps us to reduce the noise in the video by avoiding capture of unwanted movements in the surroundings.

### 4.2.2 Image pre-processing

A raw image obtained from the camera is not suitable for feature extraction or any other operations. In order to remove the noise from the background and to prepare the image for feature extraction, we utilise the techniques of image processing. One such technique is the background subtraction, which helps to differentiate between moving objects and the objects in the background.

#### 4.2.2.1 Background subtraction techniques

Background subtraction is a widely used approach for detecting moving objects in videos from static cameras. The rationale in the approach is that of detecting the moving objects from the difference between the current frame and a reference frame, often called "background image", or "background model". Background subtraction is mostly done if the image in question is a part of a video stream**.**

Two of the most widely used techniques for background subtraction are frame differencing and using Gaussian Mixture Models.

##### 4.2.2.1.1 Background subtraction using frame differencing

A motion detection algorithm begins with the segmentation part where foreground or moving objects are segmented from the background. The simplest way to implement this is to take an image as background and take the frames obtained at the time t, denoted by I(t) to compare with the background image denoted by B. Here using simple arithmetic calculations, we can segment out the objects simply by using image subtraction technique of computer vision meaning for each pixels in I(t), take the pixel value denoted by P[I(t)] and subtract it with the corresponding pixels at the same position on the background image denoted as P[B]. In mathematical equation, it is written as:

$$P[F(t)]=P[I(t)]-P[B]$$

The background is assumed to be the frame at time t. This difference image would only show some intensity for the pixel locations which have changed in the two frames. Though we have seemingly removed the background, this approach will only work for cases where all foreground pixels are moving and all background pixels are static. A threshold "Threshold" is put on this difference image to improve the subtraction.

$$|P[F(t)]-P[F(t+1)]| > \text{Threshold}$$

This means that the difference image's pixels' intensities are 'thresholded' or filtered on the basis of value of Threshold. The accuracy of this approach is dependent on speed of movement in the scene. Faster movements may require higher thresholds.

#### 4.2.2.1.2 Background subtraction using Gaussian Mixture Models

Background modeling by Gaussian mixtures is a pixel based process. Let x be a random process representing the value of a given pixel in time. A convenient framework to model the probability density function of x is the parametric Gaussian mixture model where the density is composed of a sum of Gaussians. Let p(x) denote the probability density function of a Gaussian mixture comprising K component densities:

$$p(\mathbf{x}) = \sum_{k=1}^{K} w_k \mathcal{N}(\mathbf{x}; \mu_k, \sigma_k), \qquad (1)$$

wherew$_k$ are the weights, and N (x; $\mu_k$, $\sigma_k$) is the normal density of mean $\mu_k$ and covariance matrix $\Sigma_k = \sigma_k I$, (I denotes the identity matrix). The mixture of Gaussians algorithm estimates these parameters over time to obtain a robust representation of the background. First, the parameters are initialized with $w_k = w_0$, $\mu_k = \mu_0$ and $\sigma_k = \sigma_0$. If there is a match, i.e.

$$\|\mathbf{x} - \mu_j\|/\sigma_j < \tau \text{ for some } j \in [1..K], \qquad (2)$$

where τ (> 0) is some threshold value, then the parameters of the mixture are updated as follows:

$$w_k(t) = (1-\alpha)\, w_k(t-1) + \alpha\, M_k(t), \qquad (3)$$

$$\mu_k(t) = (1-\beta)\mu_k(t-1) + \beta\,\mathbf{x}, \qquad (4)$$

$$\sigma_k^2(t) = (1-\beta)\,\sigma_k^2(t-1) + \beta\,||(\mathbf{x} - \mu_k(t))||^2, \qquad (5)$$

where $M_k(t)$ is equal to 1 for the matching component j and 0 otherwise. If there is no match, the component with the lowest weight $w_k$ is re-initialized with $w_k = w_0$, $\mu_k = x$ and $\sigma_k = \sigma_0$. The learning rate α is constant and β is defined as:

$$\beta = \alpha\,\mathcal{N}(\mathbf{x}; \mu_k, \sigma_k)\,. \qquad (6)$$

Finally, the weights $w_k$ are normalized at each iteration to add up to 1. Stauffer and Grimson proposed to sort the Gaussians by decreasing weight-to-standard-deviation ratio, $w_k/\sigma_k$, to represent the background. A threshold λ is applied to the cumulative sum of weights to find the set {1...B} of Gaussians modeling the background, defined as:

$$B = \operatorname{argmin}_{K_B} \left( \sum_{k=1}^{K_B \leq K} w_k > \lambda \right) . \qquad (7)$$

Intuitively, Gaussians with the highest probability of occurrence, $w_k$, and lowest variability in the distribution, measured by $\sigma_k$, indicating a representative mode, are the most likely to model the background.

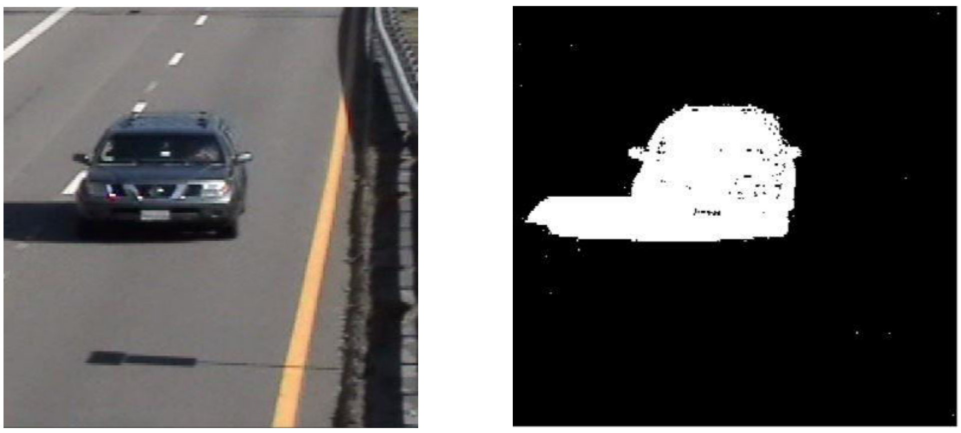


**Fig 4.2:** Image and its equivalent background subtracted image

### 4.2.3 Feature extraction

After the image has been pre-processed to remove unwanted noise and the objects of interest are identified, we can extract the required features of the objects, such as their area, centroid, Euler number etc. that may be of our interest. Here, we have chosen the area of the objects to be the feature which shall be passed as vector to the neural network.

### 4.2.4 Multilayer perceptron neural network

A multilayer perceptron (MLP) is a feed-forward artificial neural network model that maps sets of input data onto a set of appropriate outputs. An MLP consists of multiple layers of nodes in a directed graph, with each layer fully connected to the next one. Except for the input nodes, each node is a neuron (or processing element) with a nonlinear activation function. MLP utilizes a supervised learning technique called backpropagation for training the network. MLP is a modification of the standard linear perceptron and can distinguish data that is not linearly separable.

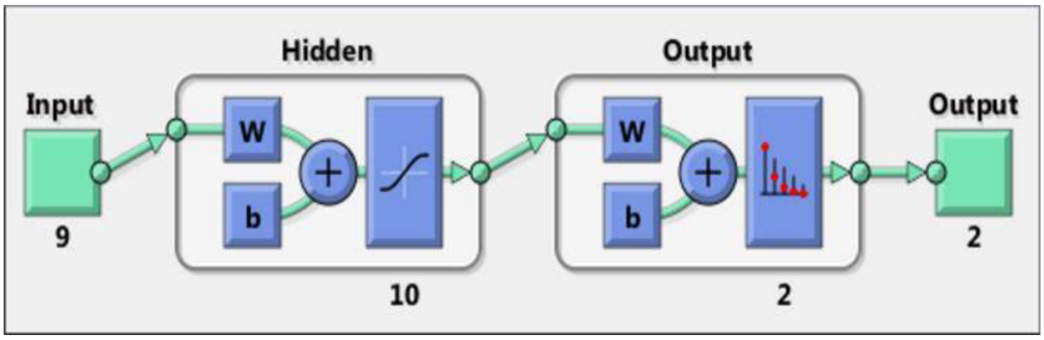


**Fig 4.3:** A sample neural network

### 4.2.4.1 Activation function

If a multilayer perceptron has a linear activation function in all neurons, that is, a linear function that maps the weighted inputs to the output of each neuron, then it is easily proven with linear algebra that any number of layers can be reduced to the standard two-layer input-output model. What makes a multilayer perceptron different is that some neurons use a nonlinear activation function which was developed to model the frequency of action potentials, or firing, of biological neurons in the brain. This function is modeled in several ways.

The two main activation functions used in current applications are both sigmoids, and are described by

$$y(v_i) = \tanh(v_i) \text{ and } y(v_i) = (1 + e^{-v_i})^{-1},$$

in which the former function is a hyperbolic tangent which ranges from -1 to 1, and the latter, the logistic function, is similar in shape but ranges from 0 to 1. Here $y_i$ is the output of the i th node (neuron) and $v_i$ is the weighted sum of the input synapses. Alternative activation functions have been proposed, including the rectifier and softplus functions. More specialized activation functions include radial basis functions which are used in another class of supervised neural network models.

### 4.2.4.2 Layers

The multilayer perceptron consists of three or more layers (an input and an output layer with one or more hidden layers) of nonlinearly-activating nodes and is thus considered a deep neural network. Since an MLP is a Fully Connected Network, each node in one layer connects with a certain weight $w_{ij}$to every node in the following layer. Some people do not include the input layer when counting the number of layers and there is disagreement about whether $w_{ij}$ should be interpreted as the weight from i to j or the other way around.

### 4.2.4.3 Learning through back-propagation

Learning occurs in the perceptron by changing connection weights after each piece of data is processed, based on the amount of error in the output compared to the expected result. This is an example of supervised learning, and is carried out through back-propagation, a generalization of the least mean squares algorithm in the linear perceptron.

We represent the error in output node j in the nth data point (training example) by $e_j(n)=d_j(n)-y_j(n)$, where d is the target value and y is the value produced by the perceptron. We then make corrections to the weights of the nodes based on those corrections which minimize the error in the entire output, given by

$$y(v_i) = \tanh(v_i) \text{ and } y(v_i) = (1 + e^{-v_i})^{-1},$$

Using gradient descent, we find our change in each weight to be

$$\mathcal{E}(n) = \frac{1}{2}\sum_j e_j^2(n).$$

wherey$_i$ is the output of the previous neuron and eta  is the learning rate, which is carefully selected to ensure that the weights converge to a response fast enough, without producing oscillations.

The derivative to be calculated depends on the induced local field $v_j$, which itself varies. It is easy to prove that for an output node this derivative can be simplified to

$$\Delta w_{ji}(n) = -\eta \frac{\partial \mathcal{E}(n)}{\partial v_j(n)} y_i(n)$$

is the derivative of the activation function described above, which itself does not vary. The analysis is more difficult for the change in weights to a hidden node, but it can be shown that the relevant derivative is

$$-\frac{\partial \mathcal{E}(n)}{\partial v_j(n)} = \phi'(v_j(n)) \sum_k -\frac{\partial \mathcal{E}(n)}{\partial v_k(n)} w_{kj}(n).$$

This depends on the change in weights of thek$^{th}$ nodes, which represent the output layer. So to change the hidden layer weights, we must first change the output layer weights according to the derivative of the activation function, and so this algorithm represents a back-propagation of the activation function.

# DESIGN AND IMPLEMENTATION

**CHAPTER 5**

# DESIGN AND IMPLEMENTATION

The architecture described in the previous section gave a high-level explanation about how the problem is tackled. Now, let us look at the low-level details of the design and the control flow.

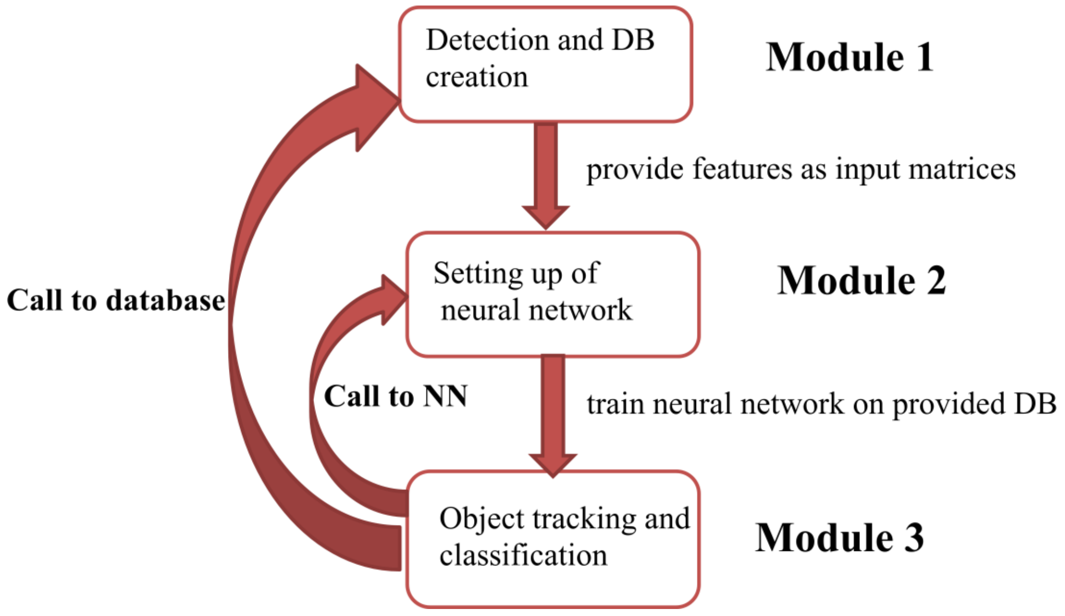
We employ a modular approach for the implementation of the application. A few points, apart from the ease of implementation, in defence of the above decision:

- The architecture is laid out such that every module, though conceptually dependent, can be implemented such that they are functionally independent.
- Also, modular approach serves as a very handy tool to keep track of the happenings at each of the functional modules.

Adhering to the above design decision, the whole structure of the program is divided into different modules. These modules are:

1. Detection and creation of the database
2. Setting up and training the neural network
3. Object tracking and vehicle classification

Let us look at the modules in detail.



Detection and DB creation
Module 1
provide features as input matrices
Call to database
Setting up of neural network
Module 2
Call to NN
train neural network on provided DB
Object tracking and classification
Module 3

**Fig 5.1:** Flowchart depicting the control flow of the modules

## 5.1 Detection and creation of database

In order to make the system intelligent, we need to provide it with some reference or learning, to detect patterns that it has not encountered before. For this purpose, it is necessary to create a robust database that can aid in the classification. To achieve this, we first pre-process the image using the background subtraction method described in previous chapters. This processed image can then be used to extract the desired feature upon which we wish to train the neural network. In our work, we have chosen the area of the blob obtained after the processing to be the input vector for the neural network. This extracted feature is stored as a column matrix, which will be the input to the neural network model.

## 5.2 Setting up and training the neural network

The neural network is set up with three layers- input, hidden, output layers. The input layer takes the matrix from the previous step for processing. The hidden layer is set up to have an optimal number of neurons, which can only be determined by observing the output of the network for some iterations. The output layer is divided to have three classes,representing the categories of vehicles that can be identified. The hidden layer is set up to have an optimal number of neurons, which can only be determined by observing the output of the network for some iterations. After the neural network is set up, it is trained on several data sets, that is, videos of several different conditions. The output is a well-trained neural network that is capable of identifying a pattern in the input, as well as detect previously unknown patterns and add to its intelligence.

## 5.3 Object tracking and vehicle classification

In order to classify vehicles in a video stream, each of the vehicles have to be detected and their features extracted and sent as an input vector to the neural network. One of the main challenges faced here is the multiple detection of a vehicle, as it appears through several frames in the video stream. This problem is overcome by using a tracking algorithm called as the Kalman filter.

### 5.3.1 Kalman Filter

Kalman filtering, also known as linear quadratic estimation (LQE), is an algorithm that uses a series of measurements observed over time, containing statistical noise and other inaccuracies, and produces estimates of unknown variables that tend to be more accurate than those based on a single measurement alone, by using Bayesian inference and estimating a joint probability distribution over the variables for each timeframe. The filter is named after Rudolf E. Kálmán, one of the primary developers of its theory.

The algorithm works in a two-step process. In the prediction step, the Kalman filter produces estimates of the current state variables, along with their uncertainties. Once the outcome of the next measurement (necessarily corrupted with some amount of error, including random noise) is observed, these estimates are updated using a weighted average, with more weight being given to estimates with higher certainty. The algorithm is recursive. It can run in real time, using only the present input measurements and the previously calculated state and its uncertainty matrix; no additional past information is required.

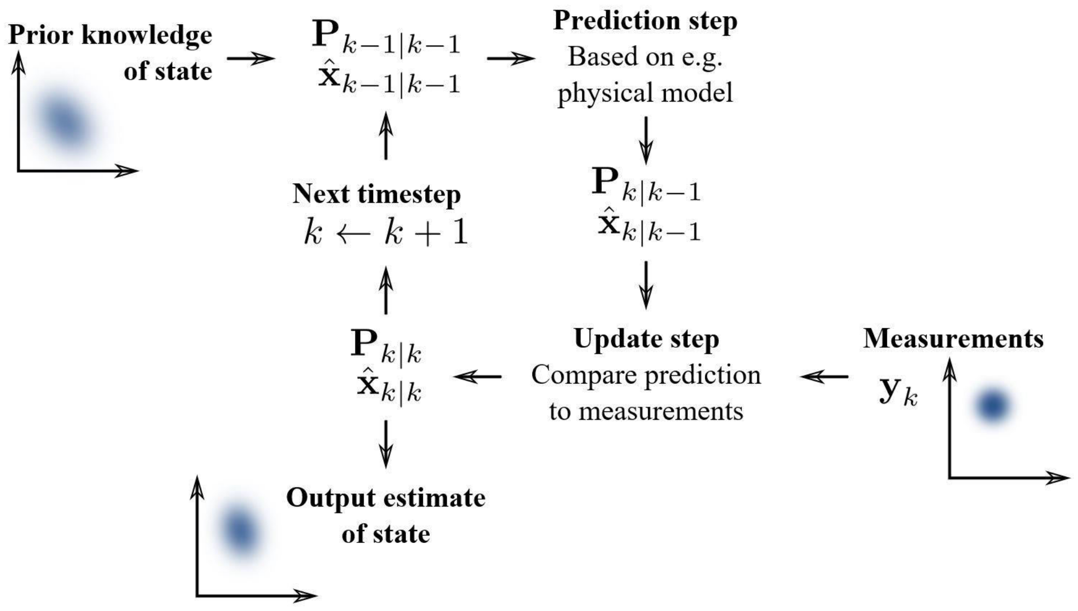

Prior knowledge of state
$\mathbf{P}_{k-1|k-1}$ $\hat{\mathbf{x}}_{k-1|k-1}$
Prediction step
Based on e.g. physical model
Next timestep
$k \leftarrow k+1$
$\mathbf{P}_{k|k-1}$ $\hat{\mathbf{x}}_{k|k-1}$
$\mathbf{P}_{k|k}$ $\hat{\mathbf{x}}_{k|k}$
Update step
Compare prediction to measurements
Measurements
$\mathbf{y}_k$
Output estimate of state

**Fig 5.2:**Working principle of Kalman filter for object tracking

The algorithm uses a structure called track to keep track of all the vehicles that have been detected. This avoids multiple detection of the same vehicle. The structure of the track is as follows:

- id - The integer id of the track.
- bbox- current bounding box of the object, used for display

- kalmanFilter -Kalman filter object used for motion based tracking.
- Age – number of frames since the track was first detected.
- totalVisibleCount – total number of frames in which the track was detected(visible).
- consecutiveInvisibleCount – number of consecutive frames in which the track was not detected(invisible).

# TESTING

# CHAPTER 6

# TESTING

To evaluate the performance of the code the code is run on different videos captured. The neural network is trained on these videos and the results have been recorded for analysis and comparison.

## 6.1 UNIT TESTING

Since our project uses the modular approach, testing each module for its correct working is called unit testing. In unit testing, the module is tested to check if it is doing what it is supposed to do. That is checking if the module is working correctly and producing the expected output. Each module, starting from creating database to classification of vehicles is producing the corresponding results. This is a conclusive proof that all units are executing without any errors.

### 6.1.1 Unit test case for creating database

| | |
|---|---|
| SL no of Test case | UTC 1 |
| Feature being tested | Database Creation |
| Sample Input | Sample Video |
| Expected Output | Matrix containing the extracted feature |
| Actual output | Matrix of extracted feature |
| Remarks | Pass |

**Table 6.1 Unit test for creating Database**

The captured video is provided as the input to create a database. The module picks random frames from the video and asks the user to enter whether they wish to add the blob to create the database. Based on the user input, the areas of the blobs are added to the database by creating a column matrix to indicate the same.

### 6.1.2 Unit test case for creating and training neural network

| | |
|---|---|
| SL no of Test case | UTC 2 |
| Feature being tested | Creating neural network |
| Sample Input | Matrix containing the extracted feature |
| Expected Output | Neural network |
| Actual output | Neural network |
| Remarks | Pass |

**Table 6.2 Creating and training neural network**

Once the database is created, a neural network with the given number of neurons is created, by taking the feature matrix as the input. It creates three output classes, corresponding to the three categories of vehicles to be classified.

### 6.1.3 Vehicle classification without creating database

| | |
|---|---|
| SL no of Test case | UTC 3 |
| Feature being tested | Vehicle classification |
| Sample Input | Sample Video |
| Expected Output | No database available |
| Actual output | No database available |
| Remarks | Pass |

**Table 6.3 Vehicle classification without database**

If we fail to create a database and try to classify the vehicles in a video stream, the program fails to do so as it has no reference to compare and learn. Hence, it cannot make a decision on the category of the detected vehicle.

## 6.2 INTEGRATION TESTING

The various modules in the project are integrated to work together to meet the common goal of the project. The correctness of this integrated assembly is tested to see if the integrity of the code is maintained. The integrity test has been carried out and found that the code produces the expected output.

## 6.3 VALIDATION TESTING

In validation testing the source code is run on different data sets and tested for its validity. As pointed out earlier the source code is run on different videos with different conditions and tested. The results show that the code is valid over different data sets, proving validation test successful.

## 6.4 SYSTEM TESTING

### 6.4.1 Vehicle classification with database

| | |
|---|---|
| SL no of Test case | UTC 4 |
| Feature being tested | Vehicle classification |
| Sample Input | Sample Video, neural network, database |
| Expected Output | Count of classified vehicles |
| Actual output | Count of classified vehicles |
| Remarks | Pass |

**Table 6.4 Vehicle classification using database**

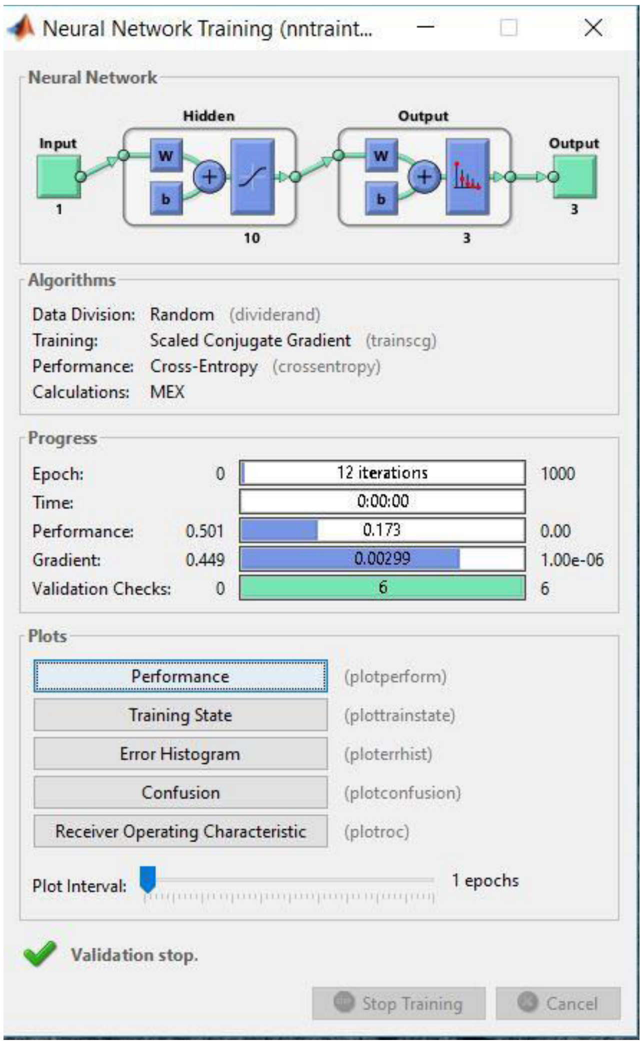
If a proper database is available, and a neural network has been trained upon this, it allows the module to effectively detect the vehicles in the video stream and classify them into one of three categories of output classes. Hence, the module provides the count of the vehicles of each class that it has detected.

# RESULTS

# CHAPTER 7

# RESULTS

The outputs of various modules have been documented here. These graphs help us to analyse the performance of the system and to make a decision about the enhancements that can be achieved further.



Algorithms

Data Division: Random (dividerand)
Training: Scaled Conjugate Gradient (trainscg)
Performance: Cross-Entropy (crossentropy)
Calculations: MEX

Progress

| | | | |
|---|---|---|---|
| Epoch: | 0 | 12 iterations | 1000 |
| Time: | | 0:00:00 | |
| Performance: | 0.501 | 0.173 | 0.00 |
| Gradient: | 0.449 | 0.00299 | 1.00e-06 |
| Validation Checks: | 0 | 6 | 6 |

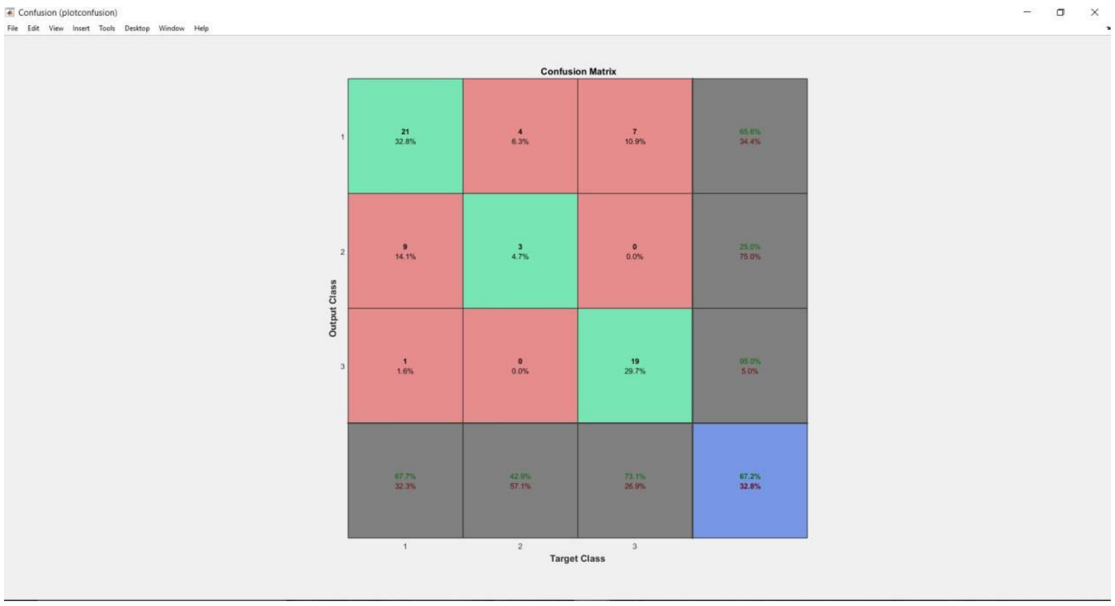
(plotperform) (plottrainstate) (ploterrhist) (plotconfusion) (plotroc)

Plot Interval: 1 epochs

Validation stop.

**Figure 7.1 Neural network training**

The above figure shows the parameters upon which the neural network is being trained. It also shows the progress achieved in the training process.


Confusion (plotconfusion)
File Edit View Insert Tools Desktop Window Help
Confusion Matrix
Output Class
Target Class

| | 1 | 2 | 3 | |
|---|---|---|---|---|
| 1 | 21<br>32.8% | 4<br>6.3% | 7<br>10.9% | 65.6%<br>34.4% |
| 2 | 9<br>14.1% | 3<br>4.7% | 0<br>0.0% | 25.0%<br>75.0% |
| 3 | 1<br>1.6% | 0<br>0.0% | 19<br>29.7% | 95.0%<br>5.0% |
| | 67.7%<br>32.3% | 42.9%<br>57.1% | 73.1%<br>26.9% | 67.2%<br>32.8% |

**Figure 7.2 Confusion Matrix**

In this figure, the first two diagonal cells show the number and percentage of correct classifications by the trained networks. Here 21 vehicles are correctly classified as light vehicles this corresponds to 32.8% of the total database. Similarly 3 are correctly classified as heavy vehicles and 19 are correctly classified as light vehicles.

4 of the light vehicles are incorrectly classified as heavy vehicles and 7 are incorrectly classified as bikes. Similarly the results can be interpreted for the other classes.

Overall 67.2% vehicles are classified correctly and 32.8% are incorrectly classified.

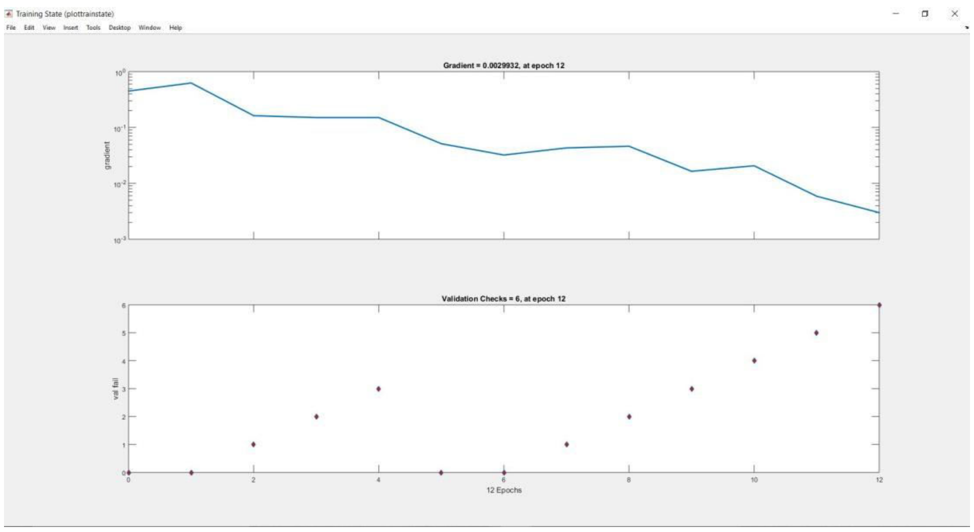


**Figure 7.3 Graph to indicate training state**

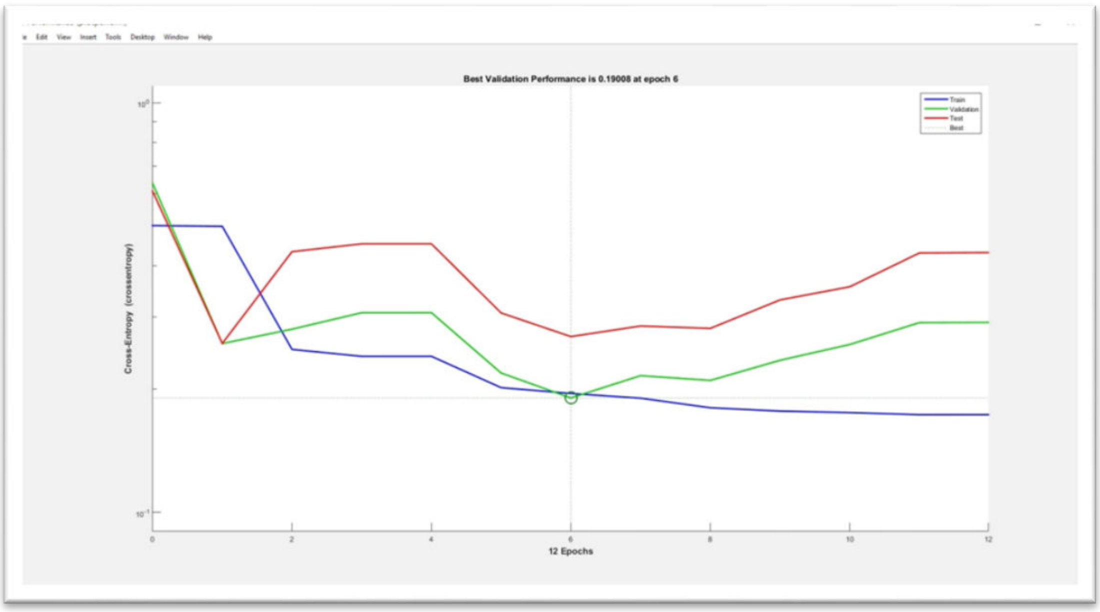


**Figure 7.4 Plot to indicate performance**

Performance plot shows the means square error dynamics for all the datasets. Gradient is the value of back propagation gradient on each iteration on logarithmic scale.

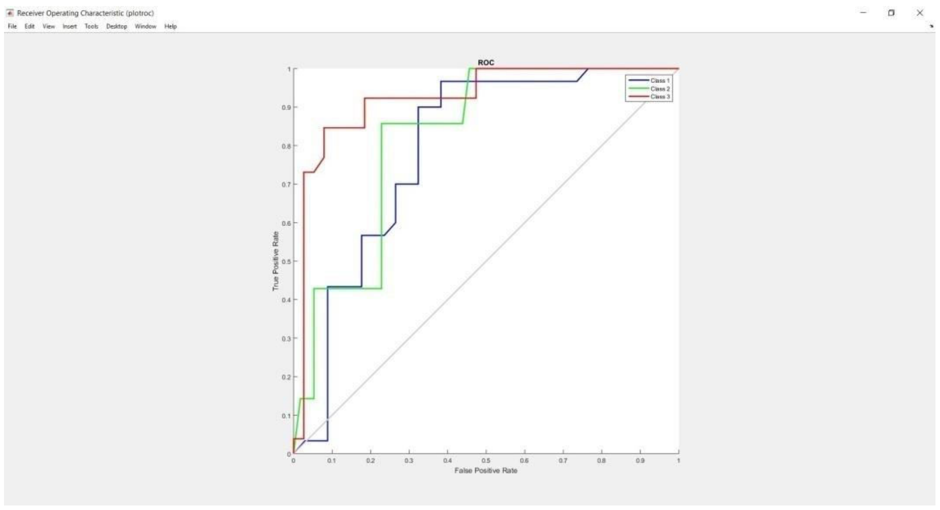


**Figure 7.5 Receiver operating characteristics**

The receiver operating characteristic is a metric used to check the quality of classifier. For each class of a classifier, ROC applies threshold values across the interval[0,1] to outputs. For each threshold two values are calculated, the True Positive Ratio (TPR) and the False Positive Ratio (FPR). The graph shows the same data as shown in figure 7.2 but in a different representation.

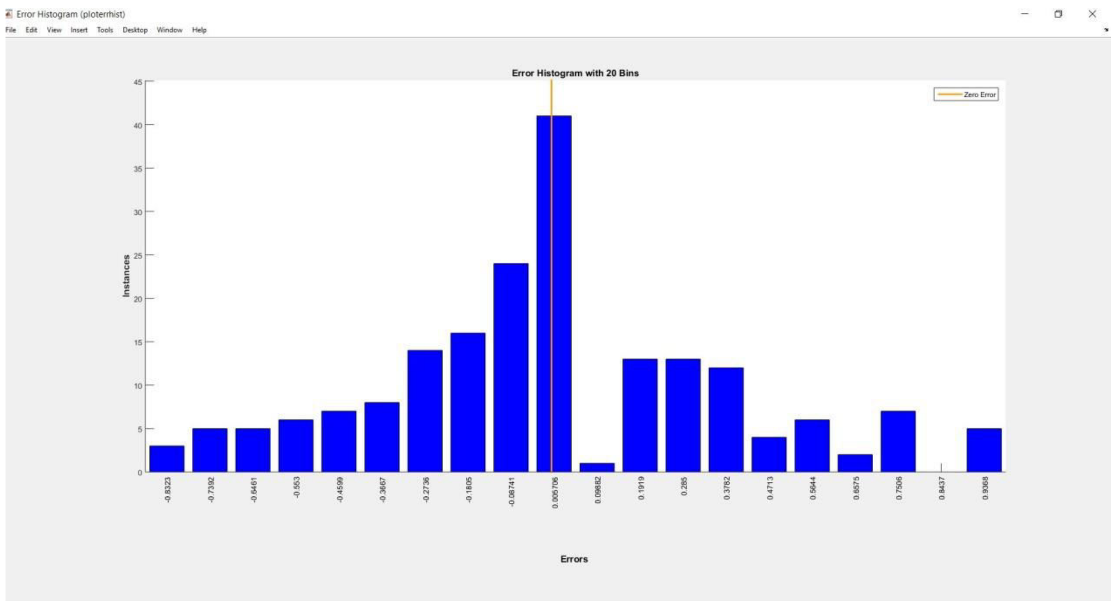


**Figure 7.6 Error Histogram**

The graph indicates the histogram of error values. It includes the number of bins that is the range of values divided into a series of intervals. The bins are usually specified as consecutive non overlapping intervals of a variable.

# CONCLUSION

# CONCLUSION

In this project, a traffic controlling system based on neural networks has been presented that it can recognize vehicles in a traffic scene and categorize it. Finding transportation parameters especially number and type of vehicles are one of the most important and applicable factors that cause to increase confidence index and reduce road crashes in roadways and highways. The camera that used in this project located in a place above the road surface. As an advantage of used method in design of algorithm is the camera don'ts need to calibration. After recognition of vehicles, they have been categorized in three classes.

This project is mainly developed for detection and classification of vehicles that can be of aid in toll booths. Several challenges have been overcome such as avoiding multiple detections of the same vehicle by adopting tracking methods, use of stationary camera and so on. The system, with very little training, is still able to perform moderately well, with a detection rate of close to 70%. The model can be greatly enhanced to not just improve its accuracy, but also to expand its applicability.

# REFERENCES

# REFERENCES

[1] K. D. Baker and G. D. Sullivan, “Performance assessment of model based tracking,” in Proc. IEEE Workshop Applications of Computer Vision, Palm Springs, CA, 1992, pp. 28–35.

[2] D.Beymer, P.McLauchlan, B. Coifman, and J.Malik, “A real-time computer vision system for measuring traffic parameters,” in Proc. IEEE Conf. Computer Visionand Pattern Recognition, PuertoRico, June1997, pp. 496–501.

[3] M. Burden and M. Bell, “Vehicle classification using stereo vision,” in Proc. 6th Int. Conf. Image Processing and Its Applications, vol. 2, 1997, pp. 881–887.

[4] A. De La Escalera, L. E. Moreno, M. A. Salichs, and J. M. Armingol, “Road traffic sign detection and classification,” IEEE Trans. Ind. Electron., vol. 44, pp. 848–859, Dec. 1997.

[5] N. Friedman and S. Russell, “Image segmentation in video sequences,” in Proc. 13th Conf. Uncertainty in Artificial Intelligence, Providence, RI, 1997.

[6] K. P. Karmann and A. von Brandt, “Moving object recognition using an adaptive background memory,” in Proc. Time-Varying Image Processing and Moving Object Recognition, vol. 2, V. Capellini, Ed., 1990.

[7] D. Koller, “Moving object recognition and classification based on recursive shape parameter estimation,” in Proc. 12th Israel Conf. Artificial Intelligence, Computer Vision, Dec. 27–28, 1993.

[8] D. Koller, J. Weber, T. Huang, G. Osawara, B. Rao, and S. Russel, “Toward robust automatic traffic scene analysis in real-time,” in Proc. 12th Int. Conf. Pattern Recognition, vol. 1, 1994, pp. 126–131.

[9] A. J. Lipton, H. Fujiyoshi, and R. S. Patil, “Moving target classification and tracking from real-time video,” in Proc. IEEE Workshop Applications of Computer Vision, 1998, pp. 8–14.

[10] O. Masoud and N. P. Papanikolopoulos, "Robust pedestrian tracking using a model-based approach," in Proc. IEEE Conf. Intelligent Transportation Systems, Nov. 1997, pp. 338–343.

[11] O. Masoud, N. P. Papanikolopoulos, and E. Kwon, "Vision-based monitoring of weaving sections," in Proc. IEEE Conf. Intelligent Transportation Systems, Oct. 1999, pp. 770–775.

[12] O. Masoud, S. Rogers, and N.P. Papanikolopoulos, "Monitoring Weaving Sections," ITS Institute, Univ. Minnesota, Minneapolis, CTS 01-06, Oct. 2001.

[13] G. D. Sullivan, "Model-based vision for traffic scenes using the ground plane constraint," Phil. Trans. Roy. Soc. (B), vol. 337, pp. 361–370, 1992.

[14] G. D. Sullivan, K. D. Baker, A. D. Worrall, C. I. Attwood, and P. M. Remagnino, "Model-based vehicle detection and classification using orthographic approximations," Image Vis. Comput., vol. 15, no. 8, pp. 649–654, Aug. 1997.

[15] G. D. Sullivan, A. D. Worrall, and J. M. Ferryman, "Visual object recognition using deformable models of vehicles," in Proc. Workshop on Context-Based Vision, Cambridge, MA, June 1995, pp. 75–86.

# SNAPSHOTS

# SNAPSHOTS

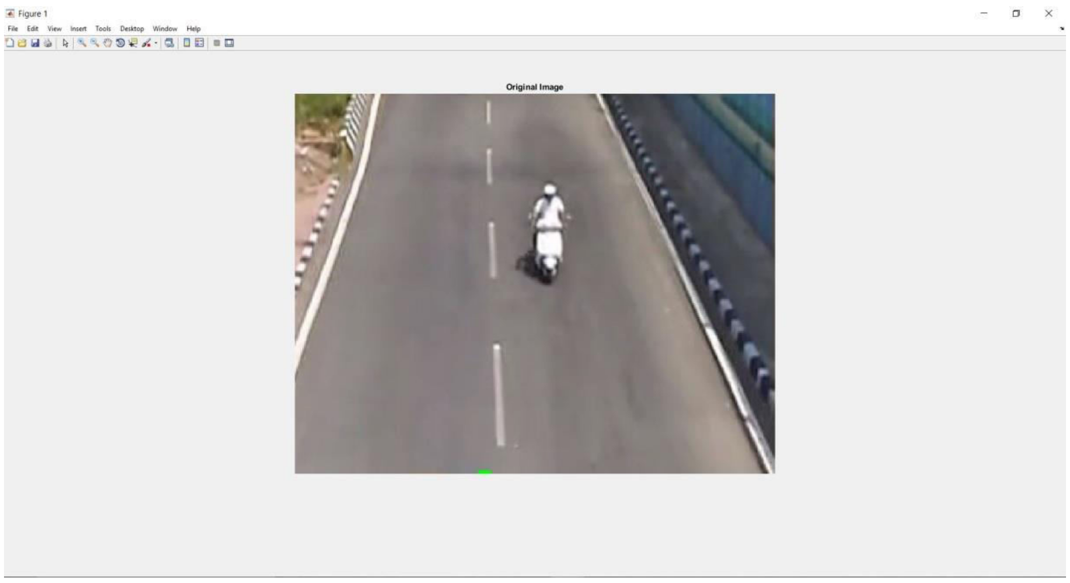


**Figure 8.1 Original Image**

The above snapshot shows the original image of a vehicle that is obtained from the traffic video sample.

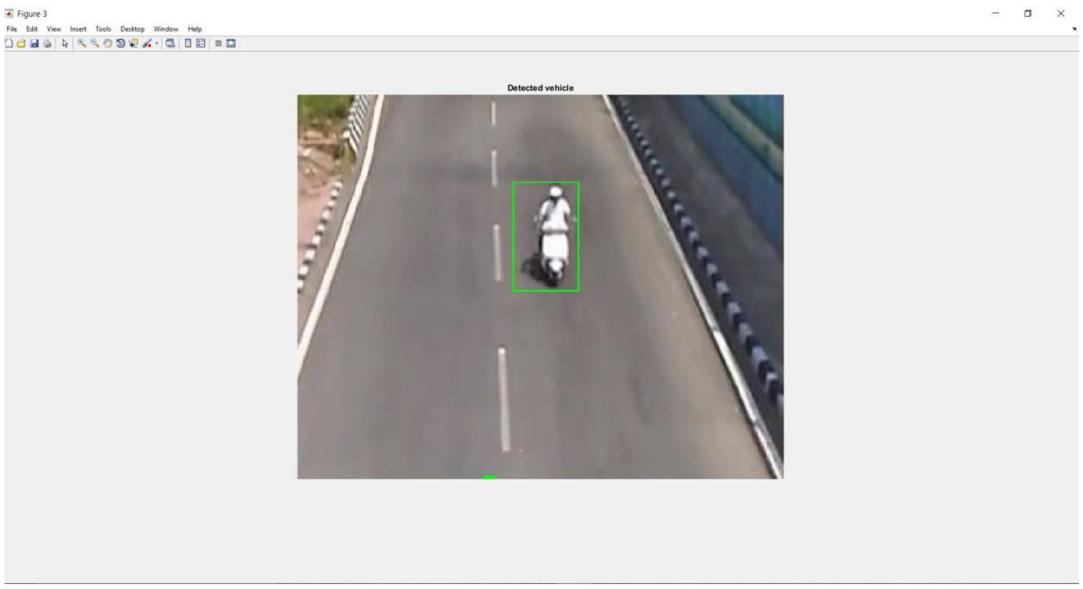


**Figure 8.2 Detected Image**

The vehicles that are present in the original image are detected. The above image shows the identified vehicle.

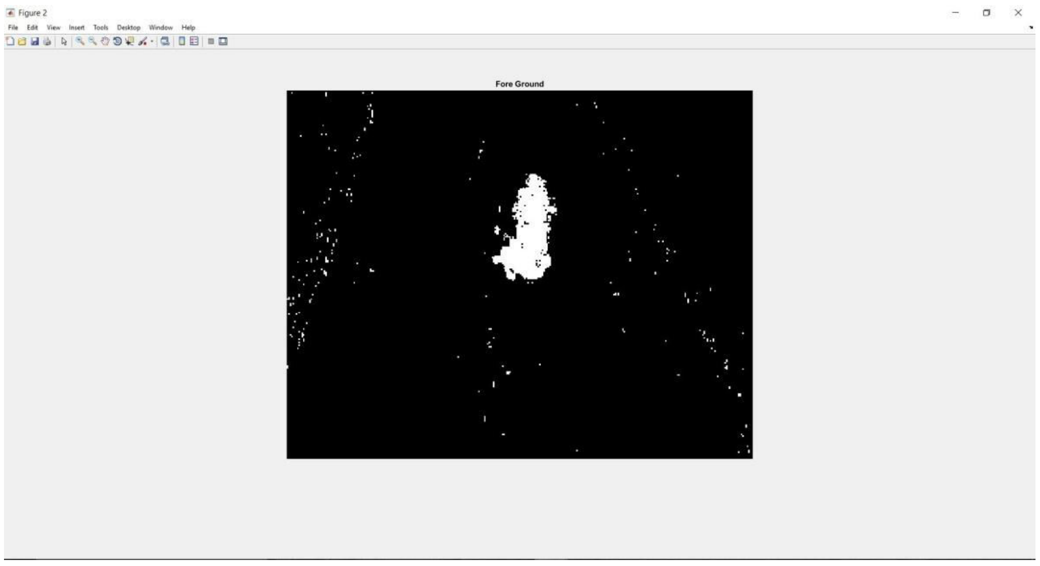


**Figure 8.3 Foreground image**

After the identification of the vehicle from the original video, the background subtraction is done and the foreground image is produced as shown in the above figure.

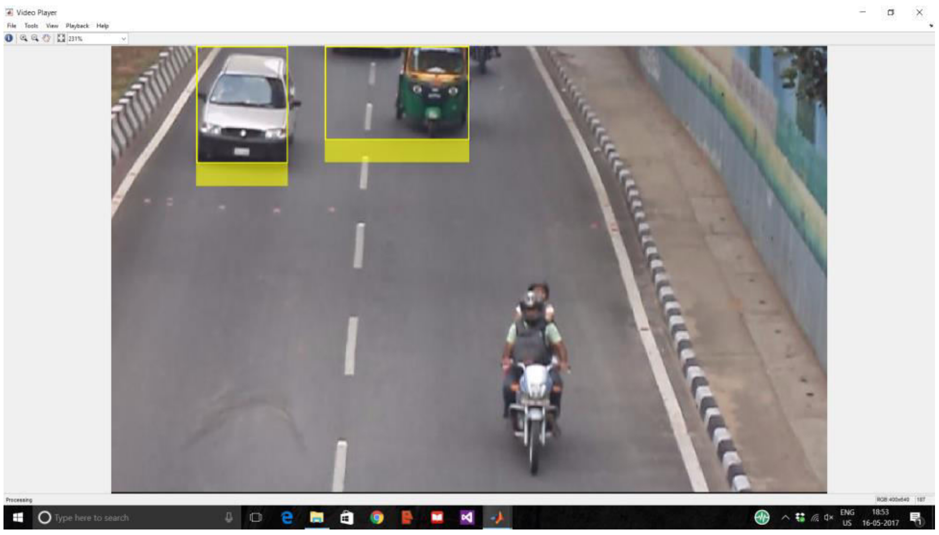


**Figure 8.4 Identified vehicles**

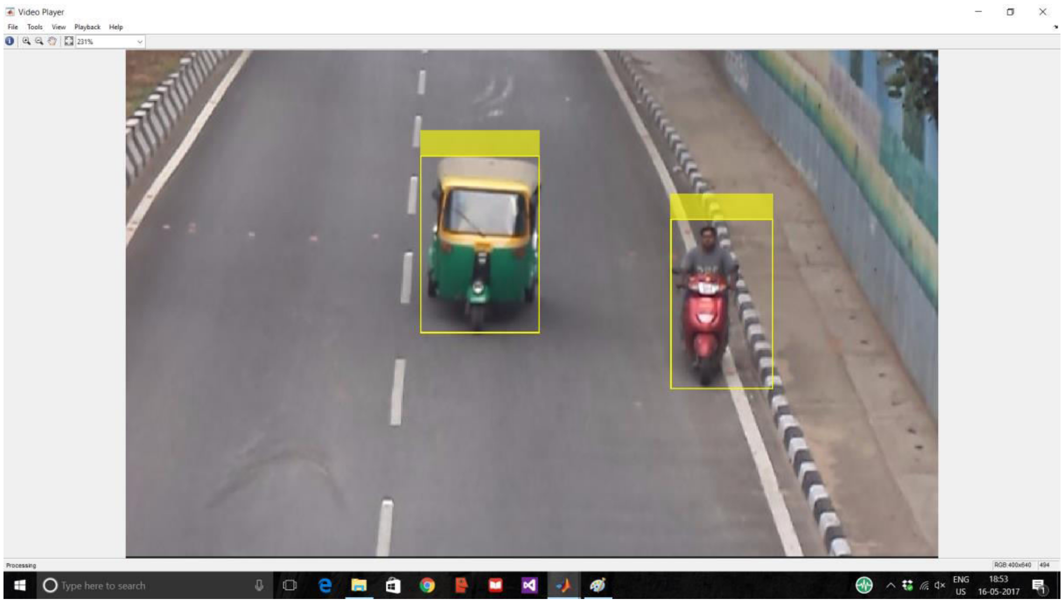


Figure 8.5 Bounding box around vehicles

The figures 8.4 and 8.5 shows the identification of multiple vehicles and the bounding box that has been drawn around the identified vehicles. This helps in keeping track of the vehicles and hence avoids multiple detection of the same vehicle.

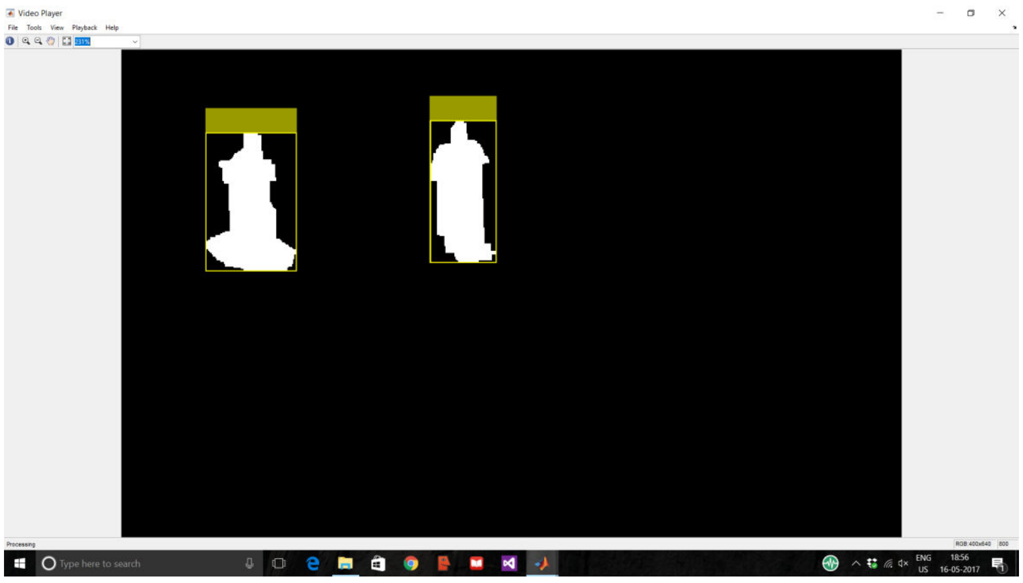


Figure 8.6 Foreground image with bounding box

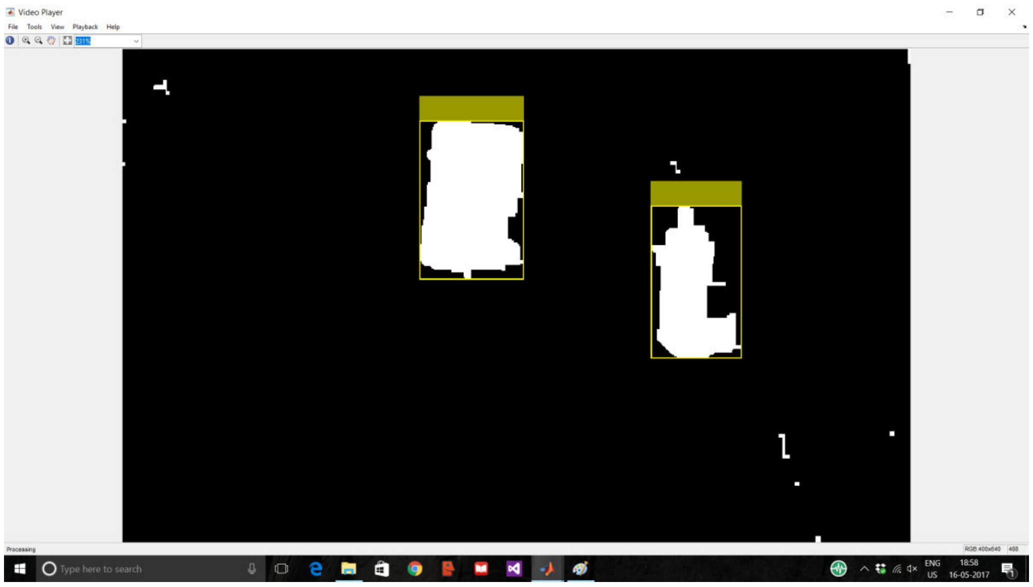


**Figure 8.7 Foreground image with bounding box**

The figures 8.6 and 8.7 shows the foreground image of the identified vehicles along with their bounding boxes around them.