VISION BASED AUTOMATIC ROAD TRAFFIC DENSITY ESTIMATION FOR BOTH STATIONARY AND FREE FLOWING TRAFFIC SCENES, AND VEHICLE CLASSIFICATION USING AN ENSEMBLE PATTERN CLASSIFIER

DANIEL ARANI OSUTO

MASTER OF SCIENCE

(Telecommunication Engineering)

JOMO KENYATTA UNIVERSITY OF AGRICULTURE AND TECHNOLOGY

2016

Vision based automatic road traffic density estimation for both stationary and free flowing traffic scenes, and vehicle classification using an ensemble pattern classifier

Daniel Arani Osuto

A thesis submitted in partial fulfilment for the degree of Master of Science in Telecommunication Engineering in the Jomo Kenyatta University of Agriculture and Technology

2016

DECLARATION

This thesis is my original work and has not been presented for a degree in any other University.

Signature …… Date ……..

Daniel Arani Osuto

This thesis has been submitted for examination with our approval as University supervisors:

Signature …… Date ……..

Prof. Heywood A. Ouma,
University of Nairobi, Kenya.

Signature ……. Date ……..

Dr. E N Ndungu,
JKUAT, Kenya.

DEDICATION

This thesis is dedicated to my parents Samuel Osuto Osuto and Nyolianah Bosibori Osuto. Your sacrifices have seen me this far. I will eternally remain thankful to you and may God bless you most abundantly.

ACKNOWLEDGEMENTS

I would like to take this opportunity to thank all who in one way or the other contributed to my research work. First, I thank my supervisors Prof. Heywood Ouma and Dr. Edward Ndungu for their support and guidance throughout the research period. They always made time for me even when they were overwhelmed with other work. I would also like to thank the Chairman, Telecommunication and Information Engineering department, Jomo Kenyatta University of Agriculture and Technology, Dr. Langat Kibet for his administrative assistance throughout the entire period that I was a student at the institution. I also acknowledge the immense support given to me by the University library staff, especially Ms Miriam Ndung'u and Ms Nanjala Barasa. They always ensured that I was supported to use the library resources optimally. Finally, I thank my classmate Kenneth Kimani Kuria, your encouragement and support, including availing the camera for the research, will not be forgotten. It kept me going even when things were very tough. May God bless you all.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF APPENDICES

LIST OF ABBREVIATIONS

Adaboost	- Adaptive boosting
ANN	- Artificial Neural Networks
Bagging	- Bootstrap aggregating
FHWA	- Federal Highway Administration
ITS	- Intelligent Transportation Systems
Kd-tree	- K-dimensional tree
KNN	- K-Nearest Neighbours
LDA	- Linear Discriminant Analysis
LoG	- Laplacian of Gaussian
NN	- Nearest Neighbour
ONVIF	- Open Network Video Interface Forum
PCA	- Principal Component Analysis
PHOG	- Pyramid Histogram of Oriented Gradients
Pixel	- Picture element
ROI	- Region of Interest
RSE	- Random Subspace Ensemble
SE	- Structuring Element
SIFT	- Scale Invariant Feature Transform
SVM	- Support Vector Machine

ABSTRACT

Automatic road traffic density estimation and vehicle classification are very important aspects of today's Intelligent Transportation Systems (ITSs). Traditionally loop sensors have been used for this purpose, but lately vision based systems have been preferred due to their advantages and the problems associated with loop sensors. Many vision based vehicle detection and classification algorithms for free flowing traffic have been proposed. These systems are largely dependent on either motion detection or more generally background modelling and subtraction. There is little reported of traffic scenes with very slowly moving or stationary vehicles for which motion detection based approaches are impractical.

This thesis presents a novel vision based road traffic density estimation and vehicle classification approach that is independent of motion detection and background modelling and subtraction. It combines selected image processing, computer vision and pattern recognition algorithms to obtain the traffic parameters. The approach is applied to standstill or slow moving traffic (more generally, 'stop - go' traffic), and free flowing traffic under different illumination conditions during daytime. The proposed system is based on a novel approach in which vehicles are simultaneously extracted from collected video frames and their image negatives through image segmentation using the Laplacian of Gaussian (LoG) edge detector and morphological filtering of the resulting binary image objects. The obtained objects are classified into small, medium and large vehicles on the basis of size using four different classification algorithms and their ensemble. The four base classifiers used are: the k-nearest neighbours (knn), the nearest centroid, the naïve Bayes and the multilayer neural

network classifiers. The proposed approach can be used for both stationary and free flowing traffic in contrast to motion detection based approaches.

To develop and test the proposed system, video data from a road section was collected using a 5 megapixel camera mounted above the road on which the subject vehicles passed. In order to assess the performance of the system under various illumination levels during the day, three datasets for the free flowing traffic were collected at three different time periods of the day. These datasets were subjected to the vehicle detection process independently. They were finally combined to form an overall dataset for the free flowing traffic which was then subjected to the pattern classification algorithms. In addition to these, a single dataset was collected from a slow moving / stationary traffic scene ('stop-go' traffic scene) and used to assess the performance of the system in such traffic scenes. The proposed algorithm was implemented in MATLAB R2015a and an average vehicle detection accuracy of 96.0% and average vehicle classification accuracies of 90.9%, 89.4%, 91.1%, 89.3% and 91.8% by the k-nearest neighbours, the nearest centroid, the naïve Bayes, the multilayer neural network and the ensemble classifiers respectively were achieved for the free flowing traffic dataset. For the slow moving traffic dataset, average detection accuracy of 82.1% and vehicle classification accuracies of 80.4%, 77.2%, 75.9%, 77.7% and 82.2% by the k-nearest neighbours, the nearest centroid, the naïve Bayes, the multilayer neural network and the ensemble classifiers respectively were achieved. The lower percentages for the later dataset were mainly due to occlusions.

The main novelty of this thesis is the development of a vision based vehicle detection algorithm that is capable of extracting vehicles from both stationary and fast moving

traffic scenes.

Chapter One

INTRODUCTION

Intelligent Transportation Systems (ITS) have become increasingly popular in today's world. Many important road traffic parameters such as traffic density and types of vehicles on the roads can be obtained automatically by these systems. One aspect of these systems that has attracted much attention among researchers in the last two decades is the use of surveillance videos to obtain the required road traffic parameters such as the traffic density and the types of vehicles on the road. Consequently, many approaches have been proposed [1]. The overwhelming majority of these approaches are dependent on motion detection or background modelling and subtraction to detect vehicles. This limits their application only to free flowing traffic scenes or scenes with static backgrounds. In cases where the traditional static background subtraction method is used, changing lighting conditions are not factored in [2], and segmentation results may thus not always be reliable. Dynamic background modelling is excellent to a large extent in handling changing scene conditions [2] [3]. Unfortunately, this method cannot be used for stationary or very slow moving traffic, a common problem in the developing world and the main focus of this research. In addition to this, many of the proposed approaches do not consider the performance in different illumination conditions. In this thesis, the background modelling and subtraction methods are avoided. Instead, a combination of simple but robust image processing and computer vision algorithms are used to extract the vehicles from the traffic video data for traffic density estimation and classification. This enables the algorithm to effectively handle stationary or very slowly moving traffic scenes.

1.1 Problem definition

In the modern world, urban centres are growing at a very high rate. Growing with these centres is the road traffic congestion. Traffic jams especially during peak hours have become routine. As a result, traffic management is one of the most pressing issues in today's towns.

One of the most viable alternatives being adopted in the developed world is the use of vision based Intelligent Transportation Systems (ITSs) to manage traffic. These systems can be used to automatically obtain critical road traffic parameters such the traffic density and the types of vehicles on the road at any given time. To obtain these parameters, the vehicles must be detected, counted and classified. To be useful, the systems must be accurate and reliable in both detection and classification of vehicles under most if not all possible traffic scenarios. Unfortunately, this is not always the case. Many of the vehicle detection algorithms that have been proposed are dependent on motion detection or more generally background modelling and subtraction. There is little reported of traffic scenes with very slowly moving or stationary vehicles, for which motion detection based approaches are impractical. Regarding vehicle classification, the 'No Free Lunch' theorems state that there is no single classifier that outperforms all others in all applications and in all circumstances. Therefore, it is difficult to select the 'best' classifier for a traffic management system from the onset and a basis for the selection is required.

1.2 Justification

High levels of accuracy and reliability are desirable attributes of any ITS. This is true especially during peak hours when road users are keen to know the traffic condition on the road ahead. Therefore, to be useful, the system must be able to handle slow moving or stationary traffic which is often the case during peak hours. Consequently, vehicle detection methods that are based on motion detection, or more generally background modelling and subtraction, cannot be useful in such scenarios. In addition to this, it is very difficult to choose the 'right' pattern classifier for a vision based ITS from the onset and a basis for the selection is required. This thesis seeks to solve these issues in two stages:

(i) developing a vehicle detection approach that can be used to effectively handle both stationary and free flowing traffic

(ii) develop a way of selecting the right pattern classifier for vision based ITSs by exploring the performance of common classification algorithms and their ensemble

1.3 Objectives

The objectives of this research were:

1.3.1 Main objective

To develop for vision based ITSs, a vehicle detection algorithm to be used for road traffic density estimation in both stationary and free flowing traffic scenes, as well as an ensemble pattern classifier for vehicle classification.

1.3.2 Specific objectives

(i) To develop a vehicle detection algorithm for use in both stationary and free flowing traffic scenes.

(ii) To design and develop individual base classifiers of the ensemble classifier algorithm.

(iii) To implement the ensemble algorithm by integrating the base classifiers.

(iv) To compare the performance of the ensemble classifier with that of its individual base classifiers and so be able to give recommendations on how to select the 'right' classifier for vision based ITSs.

1.4 Scope of the thesis

Development of an all-round vision based road traffic density estimator and vehicle classification system involves wide areas of research. Such a system will need to work not only for any possible traffic scenario, but also for any time of day (day or night). This research, however, did not consider night-time traffic. This is because night-time imaging is difficult with the use of ordinary cameras which were available for this research. The video data used for this research was collected using a non-calibrated camera, therefore, the actual areas of the traffic scenes under consideration were not computed.

1.5 Organization of the thesis

The thesis is organized as follows. Chapter Two gives overviews of the relevant image processing and pattern classification algorithms and a review of related research works. The details of the proposed algorithm are given in Chapter Three. Chapter Four presents and discusses the experimental results in detail. The conclusions of the research are drawn in Chapter Five while Chapter Six summarizes further work to be done to extend the work. The thesis concludes with the references used and appendices of materials relevant to the conducted research.

Chapter Two

LITERATURE REVIEW

In general, vision based intelligent transportation systems (ITSs) make use of image processing and pattern classification algorithms for vehicle detection and vehicle classification purposes. Overviews of these two broad areas and a review of related research works are given next.

2.1 Key relevant image processing algorithms

Image processing algorithms are at the heart of vision based ITSs. Prior to vehicle classification, the images must be appropriately pre-processed. The choice of algorithms is normally dictated by the task at hand. In this section, image processing algorithms that are central to vehicle detection and classification from the road traffic surveillance videos are briefly reviewed. These algorithms have been grouped into three subsections: image enhancement, feature extraction, and feature representation and description.

2.1.1 Image enhancement

Image enhancement is the manipulation of an image so that the output is more suitable than the original for human perception or subsequent automated processing. There is no general rule for determining the best image enhancement technique, instead, the techniques are application dependent and often developed empirically [4]. Some important image enhancement techniques relevant to this research work include: image intensity transformations and spatial domain image filtering techniques.

2.1.1.1 Intensity transformations

The image intensity transformation function at a point can be expressed as [5]:

$$s(x, y) = T[r(x, y)] \tag{2.1}$$

Where r and s are the pixel intensity values at location (x, y) for input and output images respectively. T is the transformation operator that maps r into s. Figure 2.1 [5] shows some three basic intensity transformation functions widely used for image enhancement. L is the number of intensity levels of the image, which for this research is equal to 256.

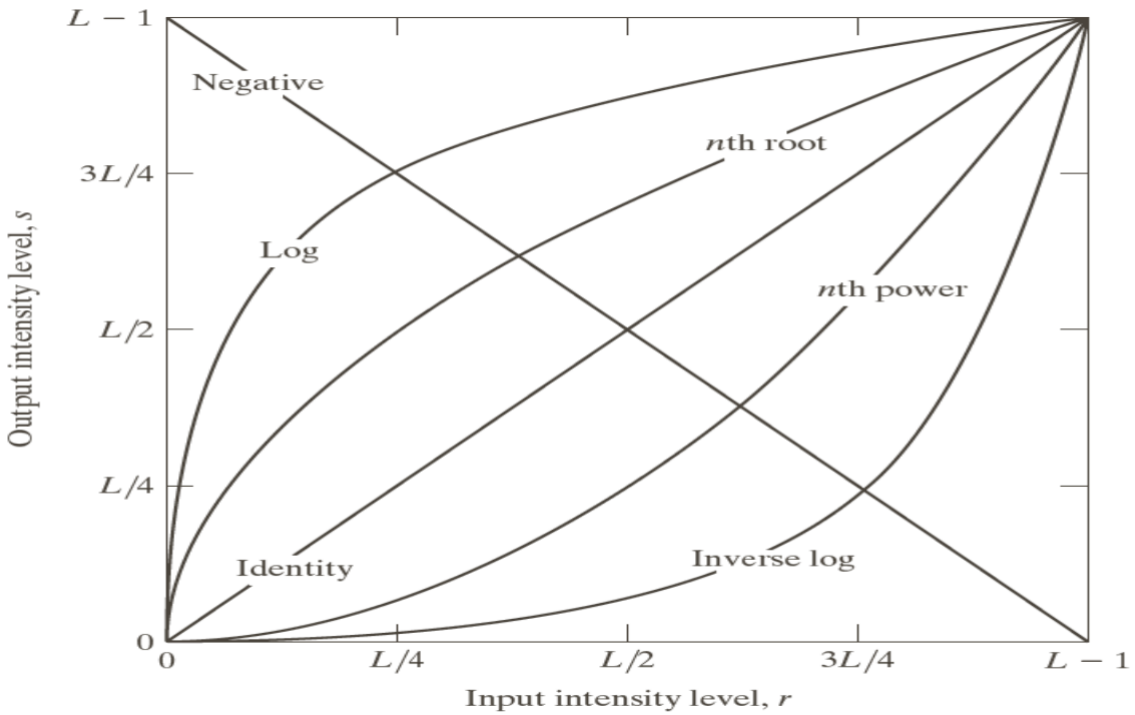


Figure 2.1: Basic intensity transformation functions

The most common image intensity transformations include: negative intensity transformation, full-scale histogram stretching transformation (contrast stretching), histogram specification (histogram shaping), histogram equalization and thresholding [5] - [10]. Thresholding is used for image segmentation while the rest are used for contrast enhancement.

Of all the listed contrast enhancement techniques, histogram equalization is the most popular due to its intuitive attributes, simplicity and good performance in problems of practical significance [5].

Histogram equalization is a non-linear mapping contrast enhancement technique that reassigns the intensity values of pixels in the input image such that the histogram of the output image is uniform [5] [7] [10]. In this way, the brightness of the pixels is distributed such that all values are equally probable [8]. It is highly popular as evidenced by the many

suggestions that have been given to improve it [11] [12]. This technique can be expressed as [5]:

$$s_k = (L-1)\sum_{j=0}^{k} p(r_j) \qquad k = 0, 1, 2...L-1 \tag{2.2}$$

Where L is the number of intensity levels in the input image, $p(r_j)$ is an estimate of the probability of occurrence of intensity level r_j, and s and r are as defined for equation (2.1)

Figure 2.2 illustrates an application of the technique in enhancing a low light image.

Figure 2.2: An illustration of histogram equalization technique

Histogram equalization is a global process and does not preserve the brightness of the image [8] [11] [12]. To mitigate this problem, a number of modifications have been proposed. Some of these include adaptive histogram equalization [5] [10] [12] [13], bi-histogram equalization [11], dualistic sub-image equalization [12], and multi-histogram equalization [11] [13] techniques.

Other than the brightness related issues, histogram equalization enhances noise along with the signal [8] and also causes unrealistic effects in images [5]. These limitations notwithstanding, this technique is simple to implement [5], and effective in detail enhancement and correction of nonlinear effects caused by a digitizer [8]. It should be noted

that a perfectly equalized histogram cannot be attained [6].

2.1.1.2 Spatial domain image filtering techniques

These refer to filtering techniques that manipulate the image pixels directly in the image plane. Spatial domain filtering can be defined by equation 2.1.

These filters are used for both image smoothing (low pass filters) and image sharpening (high pass filters).

Smoothing filters

The Gaussian, average and median filters are the typical representatives of this category. The Gaussian and average filters are linear filters while the median filter is a non-linear filter.

Gaussian filters are used for both noise removal and for blurring an image [7]. The response, R, of a linear filter mask at any given location (x, y) can be expressed as [5]:

$$R(x, y) = \sum_{k=1}^{mn} w_k z_k \tag{2.3}$$

Where the w_s are the coefficients of the normalized $m \times n$ filter mask and the z_s are the corresponding image intensities encompassed by the filter mask.

The smoothing of an image using a Gaussian filter 'removes' small objects from an image by rendering them into the background. This facilitates large object extraction from an image. The size of objects 'removed' from the image by a Gaussian blur depends on the size of the Gaussian mask used [5], which in turn depends on the magnitude of the standard deviation used [10]. The larger the mask, the greater the blurring, and the bigger the objects 'removed'. Average filters are similar to Gaussian filters except that the coefficients of their masks are all equal.

Median filters [6] are non-linear filters whose response at location (x, y) is based on ordering the pixels in the neighbourhood of (x, y) and then picking the median as the response of the filter at that location. In this way, the filter gets rid of extreme values. If such values represented noise, then the filtering will result in image improvement, otherwise it will cause image degradation. However, on average, the extreme values normally represent noise.

Median filtering is preferred to Gaussian and average filtering when excessive blurring of an image is not desirable and generally outperforms its comparable linear smoothing counterparts in removing impulse noise [5]. It also causes less blur and therefore preserves important image information such as edges and fine details [8]. Median filters can be implemented in a variety of ways [14].

In order to improve the performance of the median filters, some modifications have been proposed. The adaptive median filter [14] [15] uses variable window sizes whose dimensions change with local statistics of the image. This improves noise suppression but is computationally more complex. Fast median Filters [16] have also been proposed.

Cascaded morphological image opening and closing or vice versa can also be used for image smoothing. These filters are much simpler computationally than the median filters [6] but still yield comparable results.

Sharpening filters

First and second order image derivatives [5] [6] [17] are the most popular spatial domain sharpening filters. When combined with thresholding, these filters become edge detectors [8]. The Sobel mask is the most common of all first order image filters (image gradient filters) while the Laplacian operator is the preferred second order image derivative. Commonly used also for image sharpening is the unsharp masking technique [5].

Image gradients can also be computed using mathematical morphology algorithms [8] [15] as explained in section 2.1.2.1.

Both low pass and high pass spatial domain filters have got their corresponding frequency domain counterparts. However, spatial domain filters are much more efficient computationally [12].

2.1.2 Feature Extraction

Feature extraction is the process of obtaining distinguishing features of objects from an image. Such features form the basis for pattern classification. Features can either be intensity based or structure based. However, the extracted features should represent the objects well and be able to discriminate across the object classes with tolerance to variations [8]. There

are many ways of extracting features from an image. Some of the common ones relevant to pattern classification are mathematical morphology and image segmentation.

2.1.2.1 Morphological image processing

This is the processing of images on the basis of shape using set theory [8] [10]. In this processing, a small set called a structuring element (SE) is used to probe the active image, for properties of interest [8]. Structuring elements can take any shape but are always labelled 1 at the locations of interest and 0 elsewhere. The shape, size and origin of the SE determine the manner and extent of a morphological filtering process [5].

Common morphological filtering processes used for feature extraction and are relevant to this research include morphological erosion and dilation, morphological opening, morphological closing, skeletonization, pruning, morphological reconstruction, hole filling, and the top-hat transformation [8] [10] [15]. Erosion and dilation operations form the building blocks of all other morphological operations.

Erosion and Dilation

Erosion removes pixels from the boundaries of objects in a binary image or darkens a greyscale image. At any given location where the SE is centred, the value of the output pixel is the minimum value of all the pixels in the input pixel's neighbourhood. In a binary image, if any of the pixels is set to 0, the output pixel is set to 0 [10]. The neighbourhood is defined by the locations in the input image that overlap with the 1s of the SE. The process is repeated for all pixels in the input image. It applies to both binary and greyscale images. Erosion generally removes objects that are smaller than the SE from binary images [18]. It can be used to extract object boundaries in a binary image by subtracting the eroded image from the original image [5]. Morphological erosion is also used in the hit-or-miss transformation [8], a binary transformation commonly used to extract the convex hull and the convex deficiency [5] of objects in binary images. The convex hull and its deficiency are widely used to represent objects in classification problems.

Dilation adds pixels to object boundaries in binary images or brightens objects in greyscale images. As a result it closes the gaps between objects that are smaller than the SE (in binary images). The mechanics of dilation are the same as those for erosion. However, maximum

values rather than minimum ones are used instead. In a binary image, if any of the pixels is set to 1, the output pixel is set to 1 [10]. Iterative conditioned dilation can be used to fill holes in binary images and in the extraction of connected components from these images. Connected components are commonly used as the basis of image analysis including classification.

Erosion and dilation are also used for edge detection. The gradient of a greyscale image can be obtained using the erosion residue and the dilation residue operators. It can also be obtained as the difference between the dilated image and the eroded image [8] [15].

2.1.2.2 Image segmentation

This is the process of breaking down an image into meaningful parts having common attributes. Accuracy of this process determines the success of the classification process.

An image can be segmented using different techniques such as thresholding, edge detection, region growing and morphological watersheds [5] [6] [8] [17] [18]. Many of these techniques are usually combined to segment an image more effectively. There is no standard approach to segmentation. The choice of a given technique is dictated by the type of images and the applications [8]. Due to its simplicity of implementation and computational speed, image thresholding takes a central role in image segmentation tasks [18]. However, depending on the task at hand, other image segmentation techniques can be used.

2.1.3 Feature representation and Description

The extracted features must be represented and described in a way that will make sense to a pattern classifier. The features can be represented in many different ways such as using boundaries, signatures, skeletons and polygons. The choice of a given representation over another is dictated by the task at hand. The represented features are then described in ways such as length, area, shape numbers, texture, statistical moments and principal components [5] [18]. Again, the choice of the descriptors is determined by the task at hand.

For statistical pattern classification, these descriptions are first expressed in vector form, called feature or pattern vectors before being fed into the classifier. The classifier makes decisions based on these inputs.

2.2 An overview of Pattern classification

Pattern classification is the process of categorizing objects into distinct groups on the basis of their discriminative features. Figure 2.3 [19] shows a general classification system.

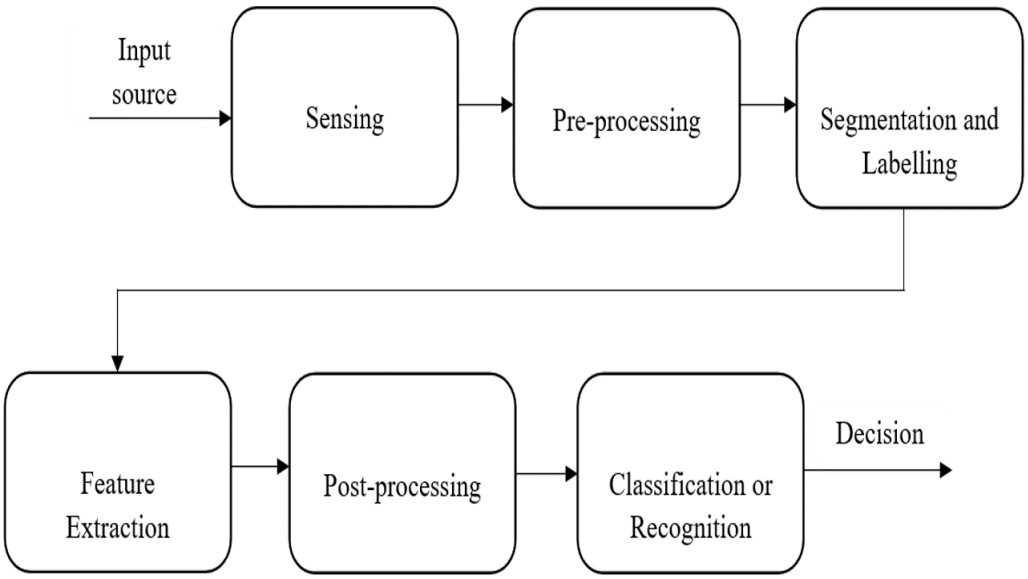


Figure 2.3: A general classification system

With respect to this research, the cameras (sensing units) collect data about the traffic scene (input source). Then, with the help of video and image processing algorithms, the data is appropriately pre-processed and the vehicles are extracted from the background using segmentation. Discriminative features of these vehicles are extracted and represented as vectors, manipulated appropriately and fed to the classifier algorithm which then makes a decision as to which class a given vehicle belongs [19] - [22]. A good overview of the stages involved in a pattern classification problem has been given in [23].

Over the years, several algorithms largely based on supervised learning have been used to classify vehicles. They include: minimum distance classifiers, Bayesian classifiers, artificial neural networks (ANN), and ensemble classifiers.

2.2.1 Minimum distance classifiers

This is one of the classifiers which are based on pattern matching, where each class is represented by prototype feature vectors and an unknown feature vector is assigned to the nearest class in terms of some metric [5]. For the minimum distance classifier, the distance between the prototype feature vectors and the unknown feature vector is computed and then the class for which this inter-feature vector distance is minimum is chosen as the class label of the unknown feature vector. Different distance metrics [24] [25] such as the Euclidean distance and the Mahalanobis distance can be used for this purpose.

The classification problem can be greatly simplified by computing the decision functions for each class and assigning the unknown feature vector to the class whose decision function yields the largest numerical value. For the minimum distance classifier, the decision function is computed as [5]:

$$d_i(\vec{x}) = \vec{x}^T \vec{g}_i - \frac{1}{2}\vec{g}_i^T \vec{g}_i \qquad \text{for} \quad i = 1, 2, ..., W \tag{2.4}$$

Where W is the number of classes, $\vec{x}$ is the unknown feature vector and $\vec{g}_i$ is the prototype feature vector representing class w_i and $d_i(\vec{x})$ is the decision function of class w_i.

There are three types of minimum distance classifiers: nearest neighbour (NN), k-nearest neighbours (KNN) and the nearest centroid minimum distance classifiers [24].

2.2.2 Naïve Bayes classifier

This is a classification algorithm which is based on the Bayes decision theory and assumes statistical independence of features within each class; but it has been shown to work well in practice even in cases where this assumption is not true [19] [25]. The classifier seeks to classify objects in such a manner that the overall probability of misclassification is minimized.

Given a set of W classes, $\{w_i\}$, for $i = 1, 2 \cdots W$ and an unknown feature vector $\vec{x}$, the feature vector $\vec{x}$ is assigned to a class with the largest posteriori probability. The conditional posteriori probability of a class w_i given the unknown feature vector $\vec{x}$ can be computed

using the Bayes formula as [22]:

$$P(w_i/\vec{x}) = \frac{p(\vec{x}/w_i)P(w_i)}{\sum_{i=1}^{W} p(\vec{x}/w_i)P(w_i)} \tag{2.5}$$

Where $p(\vec{x}/w_i)$ is the probability density function of $\vec{x}$ given that the true class to which $\vec{x}$ belongs is w_i , and $P(w_i)$ is the a priori probability of occurrence of class w_i .

The denominator is merely a normalization constant which ensures that the total probability add to 1 and since it is common to all classes, it plays no role in classification and therefore it is usually dropped in classification problems [22]. Thus, to classify the unknown feature vector $\vec{x}$, $p(\vec{x}/w_i)P(w_i)$ is computed for all classes and $\vec{x}$ is assigned to the class for which this value is largest.

In all these computations, the Bayes decision theory assumes that all information about the probabilities is known in advance. This is not always the case and an assumption has to be made. This is the main limitation of the Bayes classifier. The priori probabilities can be easily computed from the training data. However, the probability density functions are commonly assumed to be Gaussian in the literature. The performance of the classifier is affected by how well this assumption approximates reality. When this assumption is true, the Bayes classifier is optimal in the sense that the average classification error rate is minimized [23] [26].

Thus, assuming normal probability distributions for the data and a zero-one loss function for classification, the Bayes classifier decision function of class w_i becomes [5] [22]:

$$d_i(\vec{x}) = p(\vec{x}/w_i)P(w_i) \quad \text{for} \quad i = 1, 2, ..., W \tag{2.6}$$

In a logarithmic notation, the decision function is usually expressed as:

$$d_i(\vec{x}) = \ln p(\vec{x}/w_i) + \ln P(w_i) \tag{2.7}$$

In a d-dimensional space, the decision function becomes

$$d_i(\vec{x}) = \frac{1}{(2\pi)^{\frac{d}{2}}|\Sigma_i|^{\frac{1}{2}}}[-\frac{1}{2}(\vec{x} - \vec{\mu}_i)^T\Sigma_i^{-1}(\vec{x} - \vec{\mu}_i)]P(w_i) \tag{2.8}$$

Similar to equation (2.7), this decision function is usually expressed as:

$$d_i(\vec{x}) = -\frac{1}{2}(\vec{x} - \vec{\mu}_i)^T \Sigma_i^{-1}(\vec{x} - \vec{\mu}_i) - \frac{d}{2}\ln 2\pi - \frac{1}{2}\ln|\Sigma_i| + \ln P(w_i) \tag{2.9}$$

Where Σ_i and $\vec{\mu}_i$ are the covariance matrix and mean vector respectively for class w_i. Since the term $\frac{d}{2}\ln 2\pi$ is common to all classes, it can be dropped without affecting the classification results. Thus, the Bayes decision function of a class w_i for multivariate data becomes:

$$d_i(\vec{x}) = -\frac{1}{2}(\vec{x} - \vec{\mu}_i)^T \Sigma_i^{-1}(\vec{x} - \vec{\mu}_i) - \frac{1}{2}\ln|\Sigma_i| + \ln P(w_i) \tag{2.10}$$

Therefore, to classify the unknown feature vector $\vec{x}$, the Bayes decision functions for all classes are computed using equation (2.10) and $\vec{x}$ is assigned to the class whose decision function yields the largest numerical value.

2.2.3 Multilayer neural network classifier

An artificial neural network (ANN) is a mathematical model that mimics the functionality of a human nervous system. Figure 2.4 [5] shows a basic architecture of a multilayer artificial neural network.

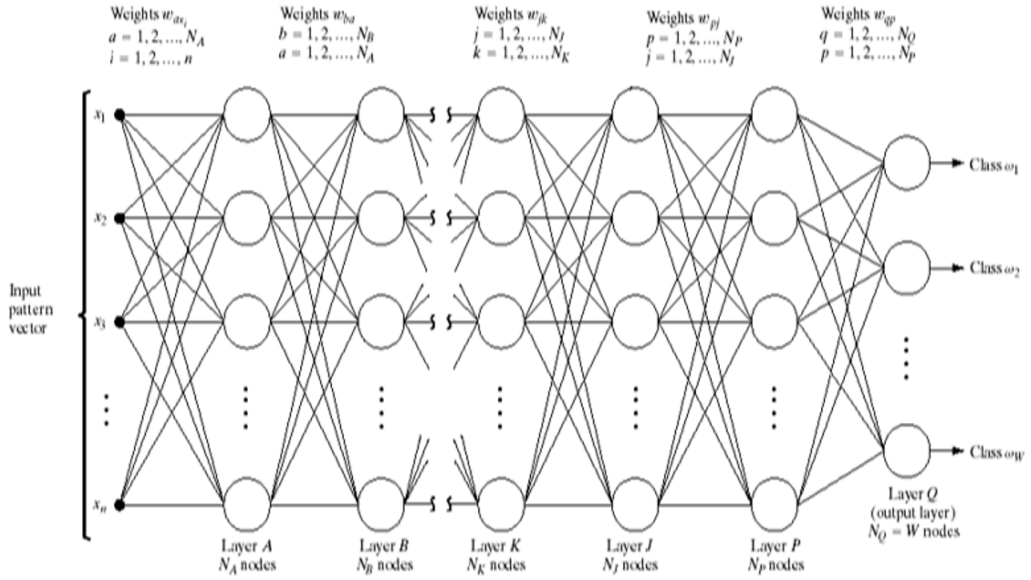


Figure 2.4: Multilayer Neural Network

The network consists of an input layer (Layer A), one or more hidden layers (Layers B... P), and an output layer (Layer Q). All the nodes (neurons) are structurally identical. They are arranged so that the output of every neuron in one layer feeds into the input of every neuron in the next layer.

An ANN is defined by the following parameters [27] [28]:

(i) Number of hidden layers in the network

(ii) Number of neurons in each layer of the network

(iii) The interconnection pattern between neurons of different layers

(iv) The activation function that converts a neuron's weighted inputs to its output activation

(v) The learning process for modifying and updating the weights of the interconnections between neurons of different layers

2.2.3.1 The number of layers and neurons per layer in a network

Any ANN with at least one hidden layer can be used for a multiclass classification problem of any complexity. It has long been established that a two layer neural network (one with one hidden layer) can implement any arbitrarily complex decision boundary (function) given a sufficient number of neurons in the hidden layer [22] [28] [29]. The number of neurons in this layer determines the complexity of the decision boundary. If this number is too small, the network will under-fit the training data and if it is too big than is necessary, the network will over-fit the training data. Neither scenario is desirable as it will result in poor generalization. Although suggestions on how to determine the right number of neurons in the hidden layer have been put forward [22] [29], in all practical cases, this number is usually determined empirically [28]. However, the number of neurons in the input layer is commonly chosen to be equal to the dimensionality of the input feature vectors while the number of neurons in the output layer is always equal to the number of classes that the neural network has been trained to recognize [5] [27] [30] [31] [32].

2.2.3.2 The activation function

In general, any function can be used as the activation function for the neurons in the network as long as such a function is nonlinear, continuous and differentiable [22] [29]. In most classification problems, the sigmoid function is used for the hidden neurons while the soft-max function is used for the output layer neurons [28].

The sigmoid function is expressed as:

$$\varphi(v) = \frac{1}{1 + e^{-v}} \tag{2.11}$$

Similarly, a soft-max function is given as:

$$\varphi(v) = \frac{e^{v}}{\sum_{i} e^{v_i}} \tag{2.12}$$

Where v is the input to the activation function.

Figure 2.5 [27] shows a dissected single neuron of the network (from Figure 2.4) that illustrates how a neural network gives an output at each neuron with the help of the activation function.

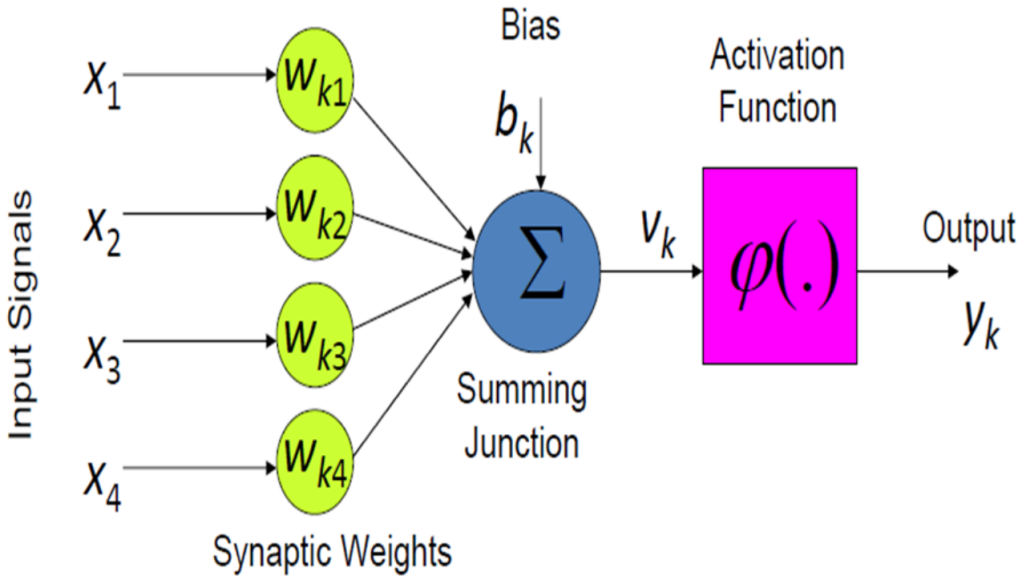


Figure 2.5: A dissected neuron

Mathematically, using the sigmoid function as the activation function, this can be expressed as:

$$y_k = \varphi(v_k) \tag{2.13}$$

$$= \frac{1}{1 + e^{-v_k}} \tag{2.14}$$

Similarly

$$v_k = \sum_{j=1}^{m} w_{kj} x_j + b_k \tag{2.15}$$

Where m is the number of weights and the rest of the terms are as defined in Figure 2.5.

2.2.3.3 Training the network: Error back propagation algorithm

Learning and updating the weights of the interconnections is an important aspect of any ANN. The training of a multilayer neural network is done by the error back propagation algorithm. The algorithm makes use of the gradient descent method to develop an error correcting learning rule. It is used for supervised training of the network. This algorithm can be summarized as follows [22] [26] [28] [29]:

At the start, the weights are randomly initialized. Then alternating forward and backward passes are used to teach the network. From the onset of the training, two values are readily available; the inputs and the desired output. The inputs are applied to the network and then the outputs are observed. The errors are computed as the differences between the actual outputs and the desired outputs. These errors are then passed backwards starting from the output layer up to the first hidden layer of the network with the help of gradient descent method, thus adjusting the weights of the network. After this, a second forward pass is applied followed by a backward pass of errors. This continues for a finite number of iterations. In a successful training, it is expected that the errors will keep decreasing with the number of iterations. However, the errors may never fall to zero, but useful estimations can be made. When the errors obtained reach a certain level, the designer can decide that the network has learned. Finally, the learned network is tested before deployment. The network recognizes a feature vector $\vec{x}$ as belonging to class w_i if the *ith* output (equation (2.14)), y_i , of the network (Figure 2.4) is high while all others are low [5]. Mathematical derivations of

this algorithm can be found in the given references.

There are many variants of the error back-propagation algorithm. However, second order based techniques are generally faster than their corresponding first order counterparts and are therefore commonly used in many practical problems [22] [26] [28] [31].

2.2.4 Ensemble classifiers

An ensemble classifier is a classifier that consists of multiple classifiers that are combined in some way to jointly give a single output. The motivation behind it is the fact that there is no single classifier that is clearly the best in all applications and in all conditions. The idea, therefore, is to combine multiple individual classifiers (base classifiers) in such a manner that they complement each other in the classification problem leading to overall improved performance. The base classifiers can be similar or dissimilar.

However, classifier combination is not "a magical formula that is always guaranteed to decrease error; base-learners should be diverse and accurate" [20]; that is, have uncorrelated errors and be good performers in their own domains of expertise.

2.2.4.1 Generating diverse base classifiers

Diverse base classifiers can be obtained by [19] [20] [21] [36]:

(i) Using different algorithms to train different base-classifiers e.g. one classifier can be KNN and another ANN.

(ii) Using different hyper parameters for similar base classifiers. For example, all the base classifiers can be ANNs but with different initial weights.

(iii) Using random subspace sampling. This entails using different subsets of classification features for different base classifiers.

(iv) Using different training sets for each base-classifier.

2.2.4.2 Classifier combination schemes

Several ways of combining outputs of the base classifiers to yield the output of an ensemble classifier have been proposed. These include: product rule, sum rule, maximum rule,

minimum rule, median rule and majority voting [20] [33] - [36]. Of all, majority voting is the most popular technique due to its simplicity and intuitive attributes [20] [23] [36].

Many algorithms for improving the performance of ensemble classifiers have been put forward. They include boosting and bagging [19] [36].

2.3 Related Work

Vision based vehicle detection and classification has long been explored. Although initial efforts were not so successful, lately much improvement has been achieved. Commercial software for this purpose exist but are dogged with problems such as inability to handle vehicle occlusions from the camera's view [37] [38], limited functionality in severe weather conditions [39] [40], shadows and night detection [41] [42]. Inter-system compatibility is one other drawback associated with today's video analytic algorithms thus severely limiting their deployment, as they do not generally work with already installed hardware unless the hardware is from the same vendor as the algorithms. Open platforms such as Open Network Video Interface Forum (ONVIF) have been formulated [43], but so far remain at the specifications stage and are not yet standardized. Each vendor understands these specifications differently and as a result, the integration of their products remains at the very basic level.

Ambardekar, *et al.* [44] and, Sivaraman and Trivedi [1] give good general overviews of the state of the art in vehicle detection and classification algorithms. In general, some of the research work done in this area focus on given problems that have long been identified in earlier works while the majority use standard algorithms to develop different approaches for detecting and classifying vehicles.

Other than the stated problems in the abstract and Chapter One, there are three main problems currently being addressed: occlusions, shadows and different weather conditions.

2.3.1 Occlusions

Vehicle occlusions have long been identified as a major bottleneck in vision based vehicle detection and classification systems. Even though many challenges still remain, good

results for handling partial occlusions have been published [37] [38] [45]. In [37], a probability-based background extraction and segmentation algorithm was used to detect partially occluded vehicles in a sequence of images by evaluating their convexity and analysing the occlusion regions before classifying them on the basis of their normalized sizes. The approach showed good ability to handle partial occlusions and classify vehicles. Pang, *et al.* [38] resolved partial occlusions between two vehicles by estimating the background using a running-average method and then used a texture-based segmentation to obtain shape contours of vehicles that were used as the basis of detecting the occlusions. The approach only failed for very severe occlusion cases. Habibu Rabiu [45] handled occlusions in cluttered urban intersections. He combined background subtraction for detection, the Kalman filter for tracking and a Linear Discriminant Analysis (LDA) classifier for classification with a good degree of success.

2.3.2 Shadows

Shadows have also been a bottleneck for vision based vehicle detection and classification systems. Shadows cause three main problems for these systems. Firstly, for free flowing traffic, the shadows move with the vehicles and the algorithm can easily 'see' them as independent vehicles or as part of the moving vehicles. Secondly, the shadows can 'join' adjacent vehicles to make bigger vehicles leading to erroneous counting and classification results; and thirdly, the shadows cause non-uniform illumination in the scene thus making segmentation difficult. Yu, *et al.* [41] proposed a vehicle tracking and classification system that takes into account size variations and shadows. Vehicle detection was achieved through background subtraction and the Kalman filter was used for tracking. The proposed shadow removal algorithm was based on the assumption that a vehicle can only be in one lane at any given time and that the distance between vehicles in adjacent lanes is uniform throughout such that vehicles in adjacent lanes can be separated optimally using a straight line. This assumption may not always hold in real world traffic scenes.

2.3.3 Different weather conditions

Severe weather conditions pose a great challenge to visual vehicle detection and classification systems. An excellent system in one condition may fail completely when

subjected to different weather conditions. This calls for systems that are capable of adapting to different weather conditions. In [46], an adaptive video-based traffic management system for counting vehicles was developed. The system was able to adapt to changing weather and illumination conditions and partially addressed the problem of occlusions. Buch, *et al.* [47] performed per frame vehicle detection and classification using 3D models under three different weather conditions: sunny condition, overcast condition and overcast changing to sunny condition. Sunny condition was reported to have classification precision results of 100% followed by overcast condition at 95.6% and overcast changing to sunny condition at 81.2%. Mishra, *et al.* [48] used background subtraction and blob tracking to detect vehicles and a kernel Support Vector Machine (SVM) classifier to classify vehicles in heterogeneous traffic scenes. Different times of the day were considered and it was shown that the performance started to deteriorate after 4.00 p.m. due to excessive reflections from the road surface. Since the system relied on motion detection, it was noted not to work for 'stop-go traffic'. Finally, Vujovic, *et al.* [49], conducted a series of experiments under different weather conditions to assess the impact of such conditions on the quality of video surveillance systems. They established that weather conditions should be considered in the development of any video based traffic management system.

From this sample of published works, it is clear that there is no single approach that works well for varied weather scenarios.

2.3.4 General approaches

The overwhelming majority of systems that have been proposed use standard image processing, computer vision and pattern recognition algorithms to handle general visual vehicle detection and classification problems. A vast majority of them are dependent on motion detection for their effective functionality, and cannot therefore be adopted for stationary traffic scenes.

Kwigizile, *et al.* [50] used probabilistic neural networks to classify vehicles according to the F-scheme [51] guidelines provided by the Federal Highway Administration (FHWA) in the USA. However, the proposed approach assumed that the probability density function of each class is known a priori. Besides, it is difficult to extract the required data for the F-scheme

classification (in which the number of axles of a vehicle are used as the basis of classification) from a natural traffic scene.

In [52], principal component analyses of front and back views of vehicles were used to classify the vehicles into either cars or trucks. Most importantly, no motion information was required to extract the vehicles, although it did not use a natural traffic scene. The proposed method assumed that the backlights of all vehicles are at the same height in the computation of "Eigen-back views" and made use of a static background in the extraction of vehicles. This assumption may not hold in a natural traffic scene.

Avery, *et al.* [42] used images from uncalibrated video cameras to count and classify trucks and heavy vehicles on the basis of length. The vehicles were extracted using a dynamic background subtraction method and then their lengths were extracted only when they reached a particular point in the scene while traveling in a straight line. In this way, reliable lengths for classification were obtained. The limitation of this approach is that it cannot work in heterogeneous traffic scenes where the vehicles are not moving in a straight line. Similarly, Pancharatnam and Sonnadara [53] used adaptive background subtraction to detect moving vehicles and tracked them using their bottom coordinates before counting and classifying them on the basis of size into large, medium and small classes. Although good results were reported, similar to [41], the system relied on the assumption that a vehicle will only occupy one lane at a time for its effective performance. It was shown that great errors occur when this is not true.

In [54], a video-based vehicle detection and classification system for real time traffic data collection using uncalibrated video cameras was proposed. The system eliminated the need for the complicated camera calibrations. It struck a good balance between algorithm complexity and effectiveness in real time applications. The paper also noted one other critical limitation of all background modelling and subtraction based algorithms (both static and dynamic) for foreground extraction: they do not account for transient lighting changes in the scene. This is confirmed by the results in [47]. Ince [55] used invariant moments and shadow aware foreground masks to count vehicles and classified them using a perspective projection of the scene geometry. The algorithm was tested on real world data and showed

to be computationally efficient.

Some researchers have also tried to emphasize the suitability of certain pattern recognition algorithms in this area while paying little attention to the necessary pre-processing stages. Cecil Ozkurt and Fatih Camci [56] proposed a system for automatic traffic density estimation and vehicle classification using neural networks and showed promising results. The system relied on dynamic background modelling for vehicle extraction and used a basic threshold for segmentation. This form of thresholding is difficult to generalize. Jain *et al.* [57] proposed a semi-supervised algorithm for estimating the traffic density from highly noisy image sources. Case studies were carried out in Nairobi, Kenya and Sao Paulo, Brazil. Both day and night times traffic were analysed. However, the thresholding approach used to segment the images rendered dark vehicles into the background and were not accounted for. In addition to this, the trial and error approach used in obtaining a suitable night time threshold value for segmentation make generalization of the algorithm difficult.

Other researchers pay more attention to the representation and description of vehicle features for classification along with the classification algorithms themselves. Ng and Tay [58] used a combination of Scale Invariant Feature Transform (SIFT), K-means clustering, and the Euclidean distance matching to develop an image-based vehicle classification algorithm. Two different vehicle classification systems were developed for two vehicle classification tasks: inter-class vehicle classification and intra-class vehicle classification. The approach hugely succeeded even though it was tested on images with single vehicles. Bailing Zhang and Chihang Zhao [59] combined Pyramid Histogram of Oriented Gradients (PHOG) and curvet transform to describe vehicles in images before classifying them into many fine classes using a Random Subspace Ensemble (RSE) of neural network classifiers. The RSE was shown to outperform individual classifiers such as the multilayer Perceptron, k-nearest neighbours and support vector machines. It was established that there are no real benefits of forming very large ensembles. Tang *et al.* [60] extracted Haar-like features from vehicles in static images and used them as inputs to a cascaded ensemble classifier. They used Adaptive Boosting (AdaBoost) to train their weak learners. The classifier was shown to attain reasonable levels of accuracy although images containing single vehicles only were considered. In [61], visual and depth information were integrated in a probabilistic

framework for vehicle detection from a moving platform with a good degree of success. The method used local features and followed a path based detection approach. Kowsari *et al.* [62] used stereo vision and multi-view Adaptive Boosting detectors for real time vehicle detection and tracking and also obtained good results.

In this thesis, other than the problems stated in Chapter One, the problem of shadows in vision based vehicle detection systems shall be addressed.

Chapter Three

THE PROPOSED ALGORITHM

In this Chapter, a multi-stage vehicle detection, counting and classification algorithm for handling both free flowing and slow moving traffic or stationary traffic is proposed. It consists of two main stages: vehicle detection and vehicle classification. It combines selected image processing and computer vision algorithms to obtain the traffic density and uses a pattern classifier to classify the vehicles into small, medium and large classes. Figure 3.1 illustrates the concept of the algorithm.

First, the collected colour video data is broken down into constituent frames which are then converted into greyscale so as to simplify and facilitate subsequent processing. The vehicles are then extracted from the video frames and their negatives using a Laplacian of Gaussian (LoG) edge detection method and mathematical morphology. The vehicles so obtained are counted and their number used to calculate the traffic density as the number of vehicles per unit area of the road section at any given time. The size dimensions of the vehicles are also extracted and used as inputs to the classifiers at the classification stage.

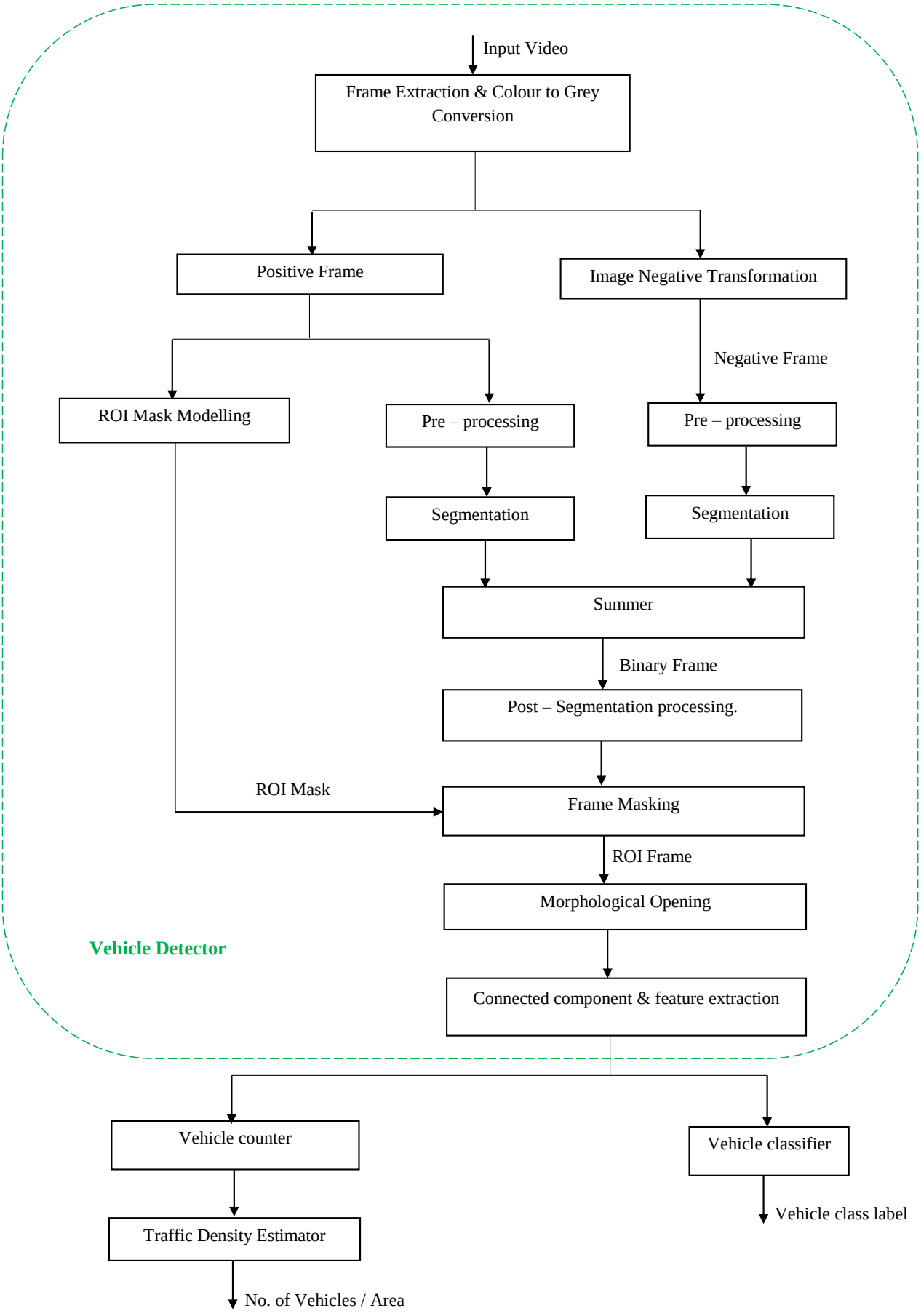


Figure 3.1: Summary of the proposed algorithm

3.1 Vehicle Detection

This is the first main stage of the algorithm. At the onset, the collected video data is broken down into individual frames for further processing. The frames are then taken through several sub – stages; the main ones of which are explained next.

3.1.1 Negative transformation

For each extracted grey frame, its image negative was computed. This was to ensure that as much relevant edge detail as possible was extracted at the segmentation stage, thus minimizing the effects of spurious edge discontinuities. In the pre-processing and segmentation sub - stages, the frame and its negative were processed independently.

The negative image of the frame was computed as [5]:

$$s = (L - 1) - r \tag{3.1}$$

Where s is the pixel value in the output image, L is the number of intensity levels in the input image and r is the pixel value in the input image. Figure 3.2 shows one of the extracted grey frames from the collected video data for free flowing traffic and its negative.

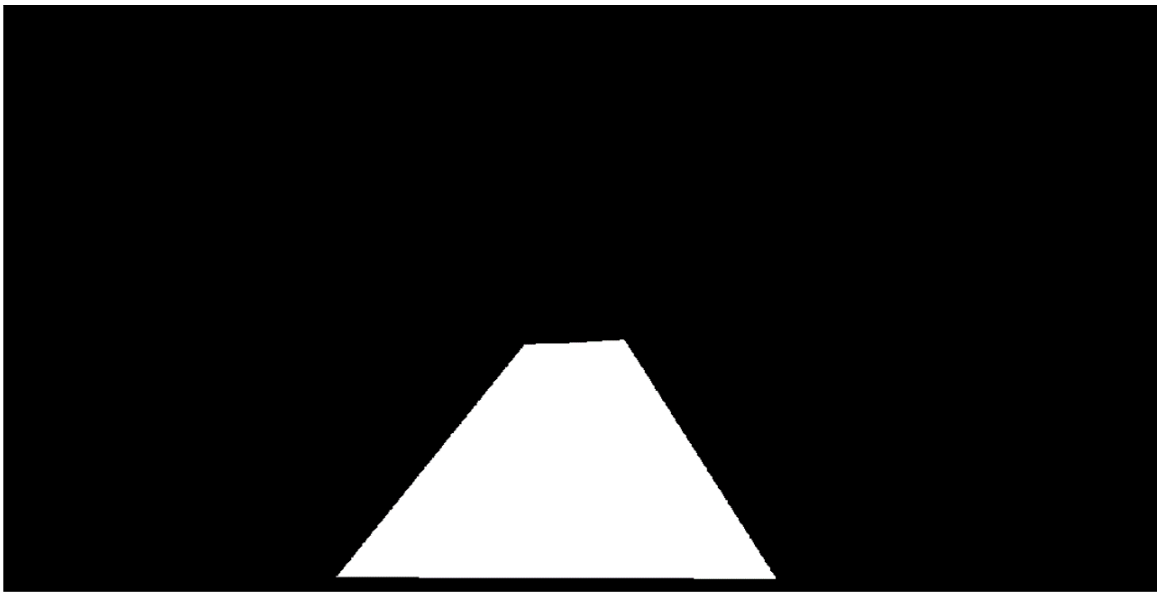
(a) Postive greyscale frame

(b) Negative image of the frame

Figure 3.2: Negative transformation

3.1.2 ROI mask modelling

One of the extracted frames was used to model manually a Region of Interest (ROI) polygon as explained in [10]. This polygon was ultimately used to mask the processed binary frames so as to limit the counting and classification of vehicles to those found only within the region of interest. The size of this polygon was chosen empirically such that the vehicle intra-class variations were minimized. Figure 3.3 shows a ROI mask modelled using the grey frame of Figure 3.2 (a) and an illustration of a masked traffic scene. However, note that frame masking in this research was deliberately brought in later for purposes of clarity in illustrating key concepts, and not at this stage as shown in Figure 3.1, and as explained in section 3.1.6. In real world deployment of the system, frame masking should come in earlier (before the pre-processing stages shown in Figure 3.1) so as to reduce computation time of the system by limiting frame processing to the ROI only.



(a) Modelled ROI mask

(b) Masked frame

Figure 3.3: ROI mask modelling

The masked frame was computed as:

$$\text{Masked frame} = (\text{Frame}) * (\text{ROI mask}) \tag{3.2}$$

3.1.3 The Top-hat transformation

The segmentation performance was improved by compensating for non-uniform illumination of the scene using the morphological top-hat transformation prior to the segmentation stage.

The top – hat transformation of the 2 – D image was computed as [10] [15]:

$$s(x,y) = r(x,y) - (r(x,y) \circ b(x,y)) \tag{3.3}$$

Where $s(x,y)$ is the output uniform background frame, $r(x,y)$ is the input frame and $(r(x,y) \circ b(x,y)$is the morphological opening of $r(x,y)$ using a structuring element (SE), $b(x,y)$. The size of the structuring element was chosen such that it was larger than any expected object of interest in the scene (i.e. the smallest expected car size) so as avoid deletion of any vehicle in the subtraction process. The top-hat transformation also helped to minimize the effects of shadows by eliminating the non-uniform backgrounds associated with shadows.

3.1.4 Image smoothing and blurring

The uniform background frame was smoothed using a median filter to remove impulse noise and then aggressively blurred using a Gaussian filter so as to render 'noise' edges into the background and therefore reduce the chances of their detection.

The 2-D discrete Gaussian filter mask was obtained by sampling the Gaussian function

$$g(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{3.4}$$

Where x and y are distances from the origin in the two dimensions and σ is the standard deviation. The magnitude of the standard deviation determines the size of the filter mask. In the proposed algorithm, a standard deviation of 5 was empirically determined and used for both free flowing and slow moving traffic scenes. Figure 3.4 shows a uniform grey frame from the previous stage, a median filtered frame and its Gaussian blurred version using a standard deviation of 5. Note from Figure 3.4 (c) that as a result of the blurring, the white markings on the road surface were rendered into the background. Similarly, the median

filtered frame looks very similar to the uniform background frame because there was no noticeable impulse noise in the original frame.

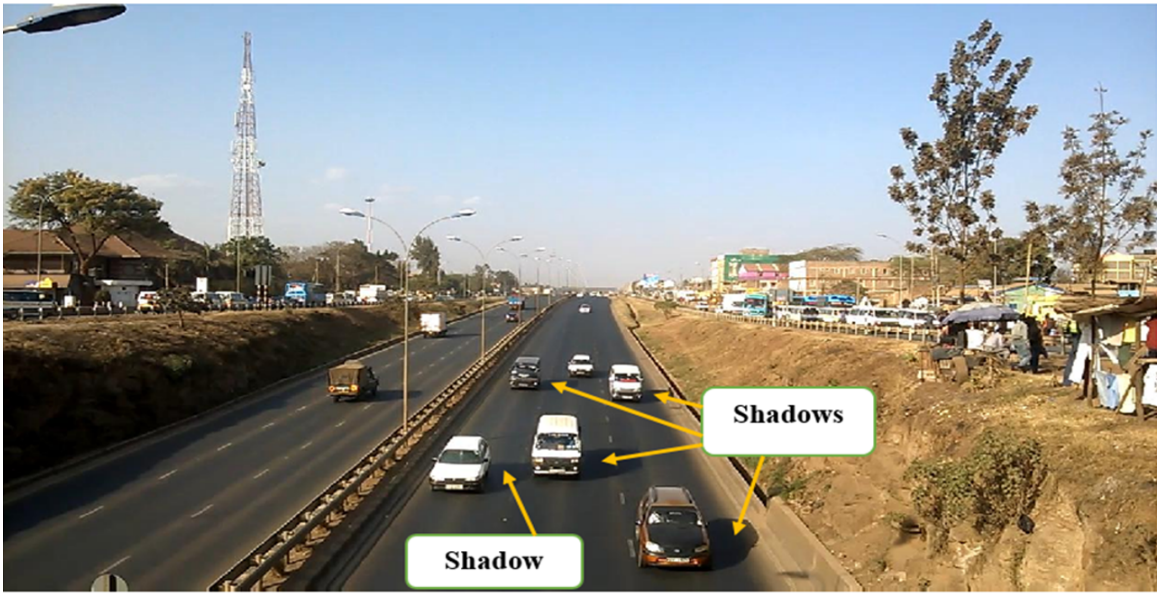
(a) Top-hat transformed frame

(b) Median filtered frame

(c) Aggressively blurred frame

Figure 3.4: Frame smoothing and blurring

Aggressive blurring also minimized the effects of shadows in the traffic scene by capitalizing on the fact that shadows are generally semi-transparent and hence through aggressive blurring, their edges closely resemble their surroundings. In this way, most parts of the edges associated with shadows were not detected at the segmentation stage (section 3.1.5), and those which were detected were not sufficient to form connected components in subsequent processing and were eventually eliminated in the post-segmentation processing stages. This approach was largely effective in handling shadows as illustrated in Figure 3.5 and subsequent sections.



(a) Original traffic scene with shadows

(b) Segmented frame with shadows removed

Figure 3.5: Shadow removal

Finally, the blurred frame was contrast enhanced linearly so as to emphasize the edges while preserving the mean intensity values of the frames using a contrast stretching algorithm prior to segmentation. The contrast stretching algorithm was implemented as [6]:

$$s = \frac{L-1}{r_{max} - r_{min}}(r - r_{min}) \tag{3.5}$$

Where r_{max} and r_{min} are the maximum and minimum input image intensity values, r and s are the pixel intensity values at location (x, y) for input and output images respectively, and L is the number of the intensity levels which is equal to 256 in this case. Figure 3.6 shows such a contrast enhanced frame of the blurred frame from Figure 3.4(c).

Figure 3.6: Contrast stretched frame

3.1.5 Image segmentation

The Laplacian of Gaussian (LoG) edge detection method was used due to its excellent edge detection properties and relative simplicity [5] [15] to extract objects in both the frame positives and their negatives. This preserved generality unlike the trial and error thresholds used in many of the approaches in literature. The LoG of a two dimensional image, $f(x, y)$ is computed as [5]:

$$\nabla^2 G(x, y) = \left[\frac{x^2 + y^2 - 2\sigma^2}{\sigma^4}\right] e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{3.6}$$

Where $\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$ is a 2-D Laplacian operator operating on a Gaussian smoothed image, $G(x, y)$, and σ is the standard deviation of the Gaussian function. Figures 3.7 and 3.8 show frames so segmented for the free flowing and slow moving traffic scenes respectively.

(a) Original colour frame

(b) Segmented frame

Figure 3.7: Frame segmentation for free flowing traffic

(a) Original colour frame

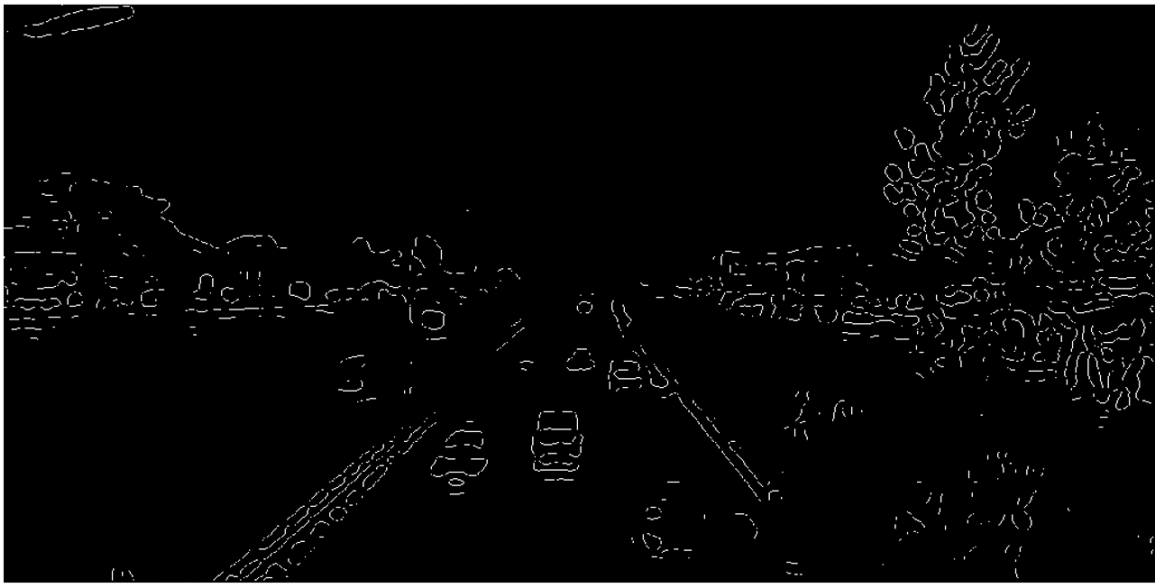
(b) Segmented frame

Figure 3.8: Frame segmentation for slow moving traffic

3.1.6 Summation and post-processing using morphological filtering

After segmentation, the positive and negative branches were added to simplify post-processing, eliminate double counts and ensure that as many objects as possible were detected. This addition is possible since the frame and its negative are spatially registered and therefore the objects which occur simultaneously in both the frame and its negative reinforce each other. The output of the summer gave the complete edge map, and therefore the binary image of the frame. Figure 3.9 shows the edge maps of the positive and negative branches of

the algorithm and their sum for the traffic scene of Figure 3.7 (a).



(a) Positive edge map

(b) Negative edge map

(c) Sum of the two edge maps.

Figure 3.9: Summation of positive and negative edge maps

Note the difference between the three edge maps. It is of interest to note that obtaining the edge map of the positive frame (original frame) alone is clearly inadequate and the huge gaps caused by spurious edge discontinuities will cause problems in subsequent stages, especially for the dark vehicles in this case, as illustrated in Figure 3.11. Computing the edge maps of both the positive and negative frames and then adding them together greatly minimized this problem as can be seen in Figure 3.9 (c).

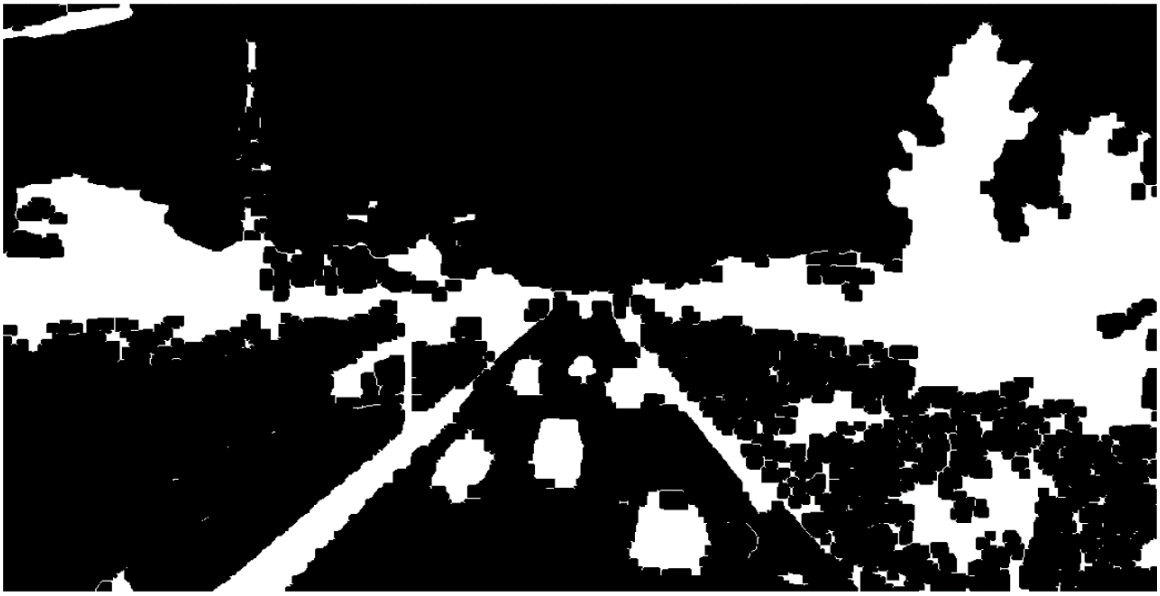
The obtained binary frame was then subjected to morphological filtering. First, the segmented binary frame was closed so as to eliminate any spurious disjoints between connected components. Then the holes in the connected components were filled to ensure that true sizes of objects were used in subsequent stages. Next, a skeletonizing algorithm was ran once before pruning the image to get rid of spurious branches that resulted after segmentation. Then, the fully processed frame was masked using the modelled ROI mask so as to limit the counting and classification to the objects found in the region of interest only, discarding everything else (see additional explanation in section 3.1.2). Finally, the irrelevant small objects such as pedestrians were deleted using a morphological opening operation. The morphological opening was done after masking the frame so as to ensure that unwanted objects on the border of the region of interest were deleted as well.

Figures 3.10 (a) – (h) illustrate these processes carrying on from Figure 3.9 (c) and making use of the ROI mask in Figure 3.3 (a). The consequence of this processing is that the shapes of the vehicles are not preserved, and therefore, cannot be used for classification. Shape preservation would be essential for intra-class classification. However, in the current inter-class classification, size dimensions alone were sufficient.

(a) Segmented frame

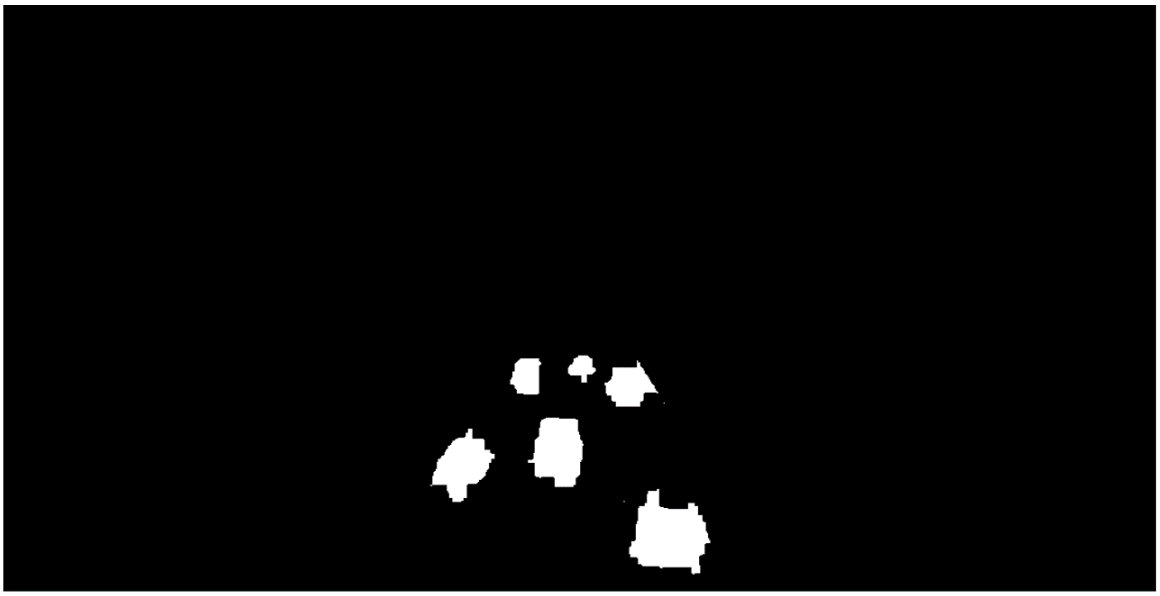
(b) Closed frame

(c) Filled frame



(d) Skeletonized frame

(e) Prunned frame



(f) Masked frame

(g) Opened ROI frame

(h) Labelled and counted objects

Figure 3.10: Post-segmentation processing

From Figure 3.10 (g), it can be seen that the vehicles were well detected and that their shadows were rendered into the background.

As explained earlier on in this section, the results obtained in Figures 3.10 (g) and 3.10 (h) would be significantly different if the negative of the frame was not used to complement the positive frame at the segmentation stage, that is, only the extracted original frame was used for subsequent processing. Figure 3.11 shows the results obtained for the same frame (Figure 3.7 (a)) processed in the same way except that the negative branch of the proposed algorithm

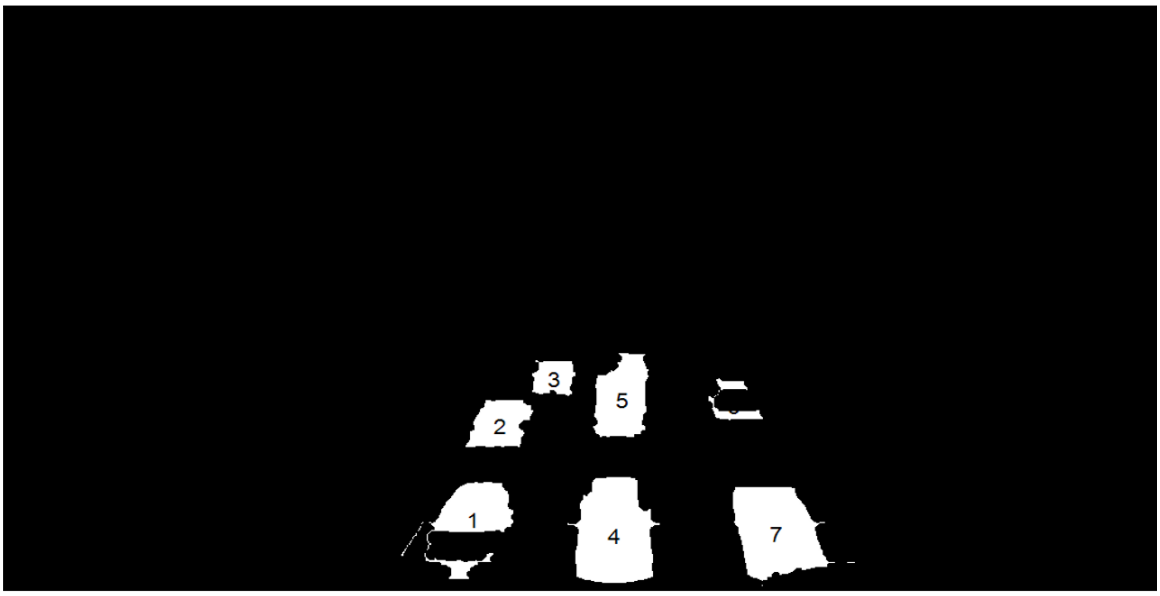
was not used.

Figure 3.11: Results for the positive frame alone

Note that object 1 in Figure 3.10 (h) has been over segmented to form objects 1 and 2 in Figure 3.11. This is due to the huge spurious discontinuities of the detected edges in the positive frame. In addition to this, objects 2 and 4 in Figure 3.10 (h) have also been eliminated from the result shown in Figure 3.11 for the same reasons. Similarly, object 6 in Figure 3.10(h) and object 5 in 3.11 represent the same vehicle. Due to the huge edge discontinuities in the positive edge map, further processing failed to take care of the discontinuities and therefore object 5 in Figure 3.11 clearly is not a good representation of the size of the vehicle in question (Figure 3.7 (a)). The net effect of all this is that the accuracies of both detection and classification of vehicles in the traffic scene will suffer as can be seen from Figure 3.11. Therefore, computing both the positive and negative edge maps of each frame and then summing them up to form the edge map of the frame improves the results obtained by the system.

Applying the same post-segmentation steps as those used in Figure 3.10 for the slow moving traffic scene of Figure 3.8 (a), the results shown in Figure 3.12 (b) were obtained. Figure 3.12 (a) shows the ROI polygon used for the slow moving traffic data.

(a) RoI polygon for the slow moving traffic data



(b) Post – segmentation processed frame

Figure 3.12: Post-segmentation processed frame for the slow moving traffic scene

Note from Figure 3.12 (b) that object 5 consists of two vehicles (Figure 3.12 (a)) but are represented as one vehicle. This is due to the fact that the car is occluded from the camera's view by the larger vehicle in front of it. This scenario highlights the importance of proper installation of cameras meant for traffic management systems.

Finally, size features of the resulting connected components were extracted and used as inputs to the classifiers at the classification stage. The classifiers assigned the vehicles appropriate class labels. The extracted features were: areas of the connected components,

major axis lengths, minor axis lengths and the areas of the bounding boxes of the connected components.

3.1.7 Vehicle counting and traffic density estimation

The resulting connected components in the processed frame represented vehicles on the road at that time. Therefore, these components were counted to give the total number of vehicles on the given section of the road at the given time. Figures 3.10 (h) and 3.12 (b) shows the results of the counts for the frames shown in Figures 3.7 (a) and 3.8 (a) respectively after processing and masking. With this value, the road traffic density can be calculated as:

$$\text{Traffic Density} = \frac{\text{Number of vehicles}}{\text{Area of traffic scene}} \tag{3.7}$$

To obtain the area of the traffic scene, camera calibration is required. This was a challenge in this research since the camera was mounted anew every time the data was to be collected; implying that the Region of Interest (ROI) was not exactly the same for all datasets even if the general traffic scene were the same (such as in the free flowing traffic scene), and therefore, camera calibration will be required each time.

3.2 Vehicle classification

This is the second main stage of the algorithm. It also involves several sub-stages. First, the extracted features were represented as vectors for each vehicle and then the vectors representing all the vehicles were concatenated into a single feature matrix before being normalized. A typical vehicle vector, $\vec{x}$, was represented as:

$$\vec{x} = [x_1, x_2, x_3, x_4]^T \tag{3.8}$$

Where x_1 is the area, x_2 is the major axis length, x_3 is the minor axis length and x_4 is the area of the bounding box of the vehicle blob.

The features were normalized using the learning data so that each feature / variable has a mean of zero and a standard deviation of 1. This was done by computing the z – scores of

each feature element in the feature matrix as:

$$z = \frac{x - \bar{x}}{\sigma} \tag{3.9}$$

Where z is the normalized feature, x is the raw feature, $\bar{x}$ is the mean of that particular feature variable and σ is its standard deviation. Normalization of the features ensured that all feature variables contributed proportionately to the classification problem and therefore prevented the classifier from having a biased preference of certain feature variables.

In this research, four primary classification algorithms, namely: the k – nearest neighbours, the nearest centroid, the naïve Bayes and the multilayer neural network classification algorithms and their ensemble were developed and tested. The vehicles were classified into small, medium and large vehicles.

3.2.1 Data acquisition

Video data from two road sections (one for free flowing traffic and the other for slow moving traffic) was collected using a 5 Megapixel camera (at a frame rate of 30 frames per second) mounted above the road on which the subject vehicles passed. To assess the performance of the system to detect vehicles under various illumination levels during the day, the data for the free flowing traffic was collected before the sun was up (0630hrs), when the sun was overhead and the shadows were negligible (1230hrs), and late in the day when both reflections from the road surface [48] and shadows are strongest (1630hrs). Data was also collected from a traffic scene that involved very slow moving traffic so as to assess the performance of the proposed system on such traffic scenes or on stationary ones. Each collection period lasted 20 minutes, resulting in at least 36000 frames each time. Note that the slow moving traffic dataset represents both the very slow moving traffic and stationary traffic since in both cases, motion detection based approaches cannot be used to extract the vehicles from the scene and an algorithm which works for one will definitely work for the other. This is critical because often during peak hours, 'stop-go' traffic is the norm. That is, vehicles are sometimes moving (slowly) and sometimes stationary, for short periods. Such was the case for the slow moving traffic dataset for this research, i.e., it involved 'stop-go' traffic.

The three video datasets of the free flowing traffic were subjected to the vehicle detection algorithm (section 3.1) independently and then the feature vectors of the extracted vehicles from the three datasets were combined to form an overall dataset for classification. Consequently, there were two datasets for classification: one for free flowing traffic and the other for slow moving traffic.

The free flowing traffic was used to assess the performance of the system under different illumination conditions during the day because its' data could be collected at any time of the day unlike for the slow moving or stationary traffic which can only be collected during peak hours.

3.2.2 Data Pre-processing: Feature selection and Feature Extraction

Feature selection is the process of deleting redundant or meaningless features of a dataset [63] while feature extraction is obtaining new and fewer more meaningful representative features through some transformation of the original feature space [19] so as to improve the speed and the generalization ability of a classifier. These reduce the dimensionality of the feature space and therefore helps to negotiate the "curse of dimensionality" [64] [65].

Dimension reduction only helps if the variables of the feature space are highly correlated. Therefore, in this work, the obtained two datasets for classification were used to determine if their four features were correlated, how they were if at all, and the extent of their correlation. To establish these, the covariance matrices of these datasets were first computed. The covariance between any two features, x and y, having means $\bar{x}$ and $\bar{y}$ respectfully for a dataset with N samples is computed as:

$$Cov(x, y) = \sum_{i=1}^{N} \frac{(x_i - \bar{x})(y_i - \bar{y})}{N - 1} \tag{3.10}$$

The computed covariance matrix of the free flowing traffic dataset is:

$$Cov\_free\_flowing = \begin{bmatrix} 1.0000 & 0.9446 & 0.9144 & 0.9555 \\ 0.9446 & 1.0000 & 0.9030 & 0.9508 \\ 0.9144 & 0.9030 & 1.0000 & 0.9320 \\ 0.9555 & 0.9508 & 0.9320 & 1.0000 \end{bmatrix} \tag{3.11}$$

Similarly, the computed covariance matrix of the slow moving traffic dataset is:

$$\textit{Cov\_slow\_moving} = \begin{bmatrix} 1.0000 & 0.8525 & 0.8284 & 0.8650 \\ 0.8525 & 1.0000 & 0.7461 & 0.9283 \\ 0.8284 & 0.7461 & 1.0000 & 0.7839 \\ 0.8650 & 0.9283 & 0.7839 & 1.0000 \end{bmatrix} \tag{3.12}$$

From the covariance matrices, it can be noted that none of the off – diagonal elements is zero, therefore, all the four features of the datasets are correlated. It can also be noted that all of these elements (in both matrices) are positive, implying that all the features are either increasing or decreasing together (i.e., changing in the same direction).

To determine the extent of the correlation, the correlation coefficient matrices are computed from the covariance matrices. The correlation coefficient between any two features is obtained by dividing their covariance by the product of their respectful standard deviations. Thus, from equation (3.10), the correlation coefficient between any two features x and y having standard deviations σ_x and σ_y is computed as:

$$\textit{Corrcoef}(x, y) = \frac{1}{\sigma_x \sigma_y} \sum_{i=1}^{N} \frac{(x_i - \bar{x})(y_i - \bar{y})}{N - 1} \tag{3.13}$$

Therefore, the correlation coefficient matrix of the free flowing traffic dataset is:

$$\textit{Corrcoef\_free\_flowing} = \begin{bmatrix} 1.0000 & 0.9446 & 0.9144 & 0.9555 \\ 0.9446 & 1.0000 & 0.9030 & 0.9508 \\ 0.9144 & 0.9030 & 1.0000 & 0.9320 \\ 0.9555 & 0.9508 & 0.9320 & 1.0000 \end{bmatrix} \tag{3.14}$$

Similarly, the correlation coefficient matrix of the slow moving traffic dataset is:

$$\textit{Corrcoef\_slow\_moving} = \begin{bmatrix} 1.0000 & 0.8525 & 0.8284 & 0.8650 \\ 0.8525 & 1.0000 & 0.7461 & 0.9283 \\ 0.8284 & 0.7461 & 1.0000 & 0.7839 \\ 0.8650 & 0.9283 & 0.7839 & 1.0000 \end{bmatrix} \tag{3.15}$$

The above correlation coefficient matrices are identical to their respective covariance matrices because the datasets were normalized.

Looking at the magnitudes of the off – diagonal elements of the correlation coefficient matrices, it is evident that all of the features were highly correlated, i.e. they largely contained similar information. It can also be noted that the correlation between the features of the free flowing traffic dataset was stronger than the one between the features of the slow moving traffic dataset. The reason for this is that occlusions were more severe for the slow moving traffic dataset resulting in more outliers in its classification dataset than for the free flowing traffic dataset (see Figure 3.12).

To obtain the amount of information that is common to any two features, their coefficient of determination was computed by squaring their correlation coefficient. Thus, squaring each element in the correlation coefficient matrices, the coefficient of determination matrix for the free flowing traffic dataset is:

$$\textit{Coefdet\_free\_flowing} = \begin{bmatrix} 1.0000 & 0.8923 & 0.8362 & 0.9129 \\ 0.8923 & 1.0000 & 0.8154 & 0.9039 \\ 0.8362 & 0.8154 & 1.0000 & 0.8687 \\ 0.9129 & 0.9039 & 0.8687 & 1.0000 \end{bmatrix} \tag{3.16}$$

Similarly, for the slow moving traffic dataset,

$$\textit{Coefdet\_slow\_moving} = \begin{bmatrix} 1.0000 & 0.7267 & 0.6862 & 0.7483 \\ 0.7267 & 1.0000 & 0.5566 & 0.8617 \\ 0.6862 & 0.5566 & 1.0000 & 0.6145 \\ 0.7483 & 0.8617 & 0.6145 & 1.0000 \end{bmatrix} \tag{3.17}$$

In percentage scales,

$$\textit{Coefdet\_free\_flowing} = \begin{bmatrix} 100 & 89.23 & 83.62 & 91.29 \\ 89.23 & 100 & 81.54 & 90.39 \\ 83.62 & 81.54 & 100 & 86.87 \\ 91.29 & 90.39 & 86.87 & 100 \end{bmatrix} \tag{3.18}$$

$$Coefdet\_slow\_moving = \begin{bmatrix} 100 & 72.67 & 68.62 & 74.83 \\ 72.67 & 100 & 55.66 & 86.17 \\ 68.62 & 55.66 & 100 & 61.45 \\ 74.83 & 86.17 & 61.45 & 100 \end{bmatrix} \tag{3.19}$$

For the free flowing traffic dataset, this means that features 1 and 4 (areas and areas of the bounding boxes respectively), for example, contain 91.29% similar information and only (100% - 91.29%) = 8.71% of information differentiates them. Similarly for the slow moving traffic dataset, features 2 and 4 (major axis lengths and areas of the bounding boxes respectively), contain 86.17% similar information and only (100% - 86.17%) = 13.83% of information differentiates them. Consequently, using these features directly the way they are will lead to unnecessarily complex classification models which will result in over–fitting. On the basis of Occam's razor [19] [22], simpler models that are sufficient for a given classification task are always preferred and generally give better out of sample performance.

After establishing that the features were correlated, an effort was made to obtain a sufficient minimum number of features for classification without losing too much information so that classification errors were minimized. To do this, the feature extraction method was used. This is because feature selection methods result in loss of too much information since entire features are discarded.

In this thesis, the Principal Component Analysis (PCA) [5] [66], a dimension reduction approach commonly used in face recognition and image compression problems was adopted for all the classification models. The PCA algorithm is a feature extraction method in which the entire feature space is linearly transformed into a new feature space of similar dimensions. However, much of the information in the old feature space is contained ('compressed') in a few dimensions in the new feature space. Therefore, the new feature space dimensions that carry little information can be discarded without losing too much information. In so doing, none of the features in the old feature space is discarded completely because every feature in the new feature space carries some information about every feature

in the old feature space.

In the PCA transformation, the new feature space consists of eigenvectors where the number of the eigenvectors is equal to the number of features in the old feature space, i.e. the two feature spaces are of equal dimensions. The eigenvalues of these eigenvectors are the variances and therefore the amount of information of the dataset in the directions of their respectful eigenvectors. Thus, by discarding the eigenvectors whose eigenvalues are negligibly small, the dimensionality of the dataset can be reduced with little loss of information.

Computing the PCA of the normalized free flowing traffic dataset, the eigenvalues of the resulting four eigenvectors in descending order were: 3.7987, 0.1045, 0.0544 and 0.0403. From these values, the first eigenvector carries 95.02% of all information about the dataset. Therefore, by discarding the other 3 eigenvectors, the 4 dimensional dataset was transformed into a 1-dimensional dataset with only 4.98% loss of information. Similarly for the slow moving traffic dataset, the eigenvalues of the resulting four eigenvectors in descending order were: 3.5043, 0.2917, 0.1347 and 0.0693. For this dataset, the first and second eigenvectors carry 87.61% and 7.29% of information respectively about the dataset. Jointly, these two eigenvectors carry 94.9% of all information about the slow moving traffic dataset. Thus, by discarding the other 2 eigenvectors, the 4 dimensional dataset was transformed into a 2-dimensional dataset with only 5.1% loss of information.

The application of the PCA algorithm led to improved classification accuracies on both datasets by an average of 1.1% for the free flowing traffic dataset and 3.9% for the slow moving traffic dataset.

3.2.3 Designing the base classification models

Through empirical means, 5-fold cross-validation technique was used for classification by the four individual classification algorithms.

In the cross-validation technique, a dataset is broken down into a number of subsets (say k) called folds. Then, these folds are used for training and testing a classifier iteratively, called k-fold cross-validation. That is, the classifier is trained and tested k times and then the

average classification accuracy of the classifier is obtained by averaging out the classification accuracies obtained in the k trials. In each trial, one fold is used for testing while all the others are used for training. The procedure is repeated until all folds have been used for testing, i.e. k times.

Cross-validation ensures that every data point in a dataset is used for both training and testing without compromising the results. This is a big advantage in classification problems especially those involving small datasets in which the data is not enough to be broken down into distinct training and testing sets (the holdout method). In addition to this, the technique eliminates the possibility of having either the training or testing set being biased and therefore yielding erroneous results, since such errors are not only averaged out, but also every point will form part of the training and testing sets at some point during the classification process.

The classifiers used in the construction of the ensemble classifier in this research are: the k-nearest neighbours, the nearest centroid, the naïve Bayes and the multilayer neural network classifiers.

3.2.3.1 The K–Nearest Neighbours classifier

In designing this classifier, each vehicle class was represented by prototype training feature vectors and an unknown vehicle feature vector was assigned to the class of the simple majority of its k-nearest neighbours across the entire dataset. To determine these neighbours, the Euclidean distances between the unknown feature vector and the prototype feature vectors of all the classes were computed. The Euclidean distance between two points $P(p_1, p_2 \cdots p_N)$ and $Q(q_1, q_2 \cdots q_N)$ in N dimensional space is calculated as:

$$d(P,Q) = \sqrt{\sum_{i=1}^{N}(p_i - q_i)^2} \tag{3.20}$$

All the distances so obtained were given equal weights, and therefore, all the k-neighbours had equal voting weights. This gave better results (by average classification accuracies of 2.5% and 2.7% for the free flowing and slow moving traffic datasets respectively) than the inverse distance based weights proposed by Dudani [67], in which the closest neighbours are given the highest weights.

In order to improve computational efficiency in searching for the required nearest neighbours, the kd-tree algorithm, a branch and bound search algorithm proposed by Friedman et al. [68] was used. At times, when computing the k–nearest neighbours, ties occur. Such ties were broken by assigning the unknown feature vector to one of the tying classes whose element was nearest to this feature vector.

In choosing the value of k (i.e. the number of neighbours), no concrete rule exists. However, the generally accepted rule of thumb is: "The number of neighbours, k is selected as a trade-off between choosing a value large enough to reduce the sensitivity to noise, and choosing a value small enough that the neighbourhood does not extend to the domain of other classes. Typically a procedure such as cross-validation is used to optimize the choice of k" [23]. Using this approach, the value of k was obtained to be ten for all the five subsets for both datasets.

3.2.3.2 The Nearest Centroid classifier

Just like the k-nearest neighbours classifier, this is also a minimum distance classifier and its design is the same as that of the k-nearest neighbours except that each class is represented by the mean vector of its training prototype samples rather than the training samples themselves and the value of k is always equal to 1. Therefore, feature space distances were computed between these mean vectors and the unknown feature vector, and then the unknown feature vector was assigned to the class of the nearest centroid (mean vector).

3.2.3.3 The Naïve Bayes classifier

The naïve Bayes classifier seeks to classify objects in such a manner that the overall probability of misclassification is minimized as explained in section 2.2.2. Given a set of the three classes, $\{w_i\}$, for $i = 1, 2, 3$ and an unknown feature vector $\vec{x}$; the Bayes decision functions (equation (2.10)) of the three classes were computed and then $\vec{x}$ was assigned to the class whose decision function yielded the largest numerical value. The Bayes decision function for multivariate data given in equation (2.10) was used for classification, i.e.

$$d_i(\vec{x}) = -\frac{1}{2}(\vec{x} - \vec{\mu}_i)^T \Sigma_i^{-1}(\vec{x} - \vec{\mu}_i) - \frac{1}{2}\ln|\Sigma_i| + \ln P(w_i)$$

Where $\vec{x}$ is the unknown feature vector, w_i is one of the three classes, $P(w_i)$ is the priori probability of occurrence of class w_i, Σ_i and $\vec{\mu}_i$ are the covariance matrix and mean vector respectively for class w_i. The priori probabilities of the three classes in the two datasets were computed from the training data.

Since after the application of the PCA algorithm the new feature space of the free flowing traffic dataset became one dimensional, the covariance matrix in the given decision function is simply the variance of the data in the new feature space for this dataset.

3.2.3.4 The Multilayer Neural Network classifier

The ANN design parameters explained in section 2.2.3 were addressed as follows:

The number of layers and neurons per layer in the network

As explained in section 2.2.3, a two layer neural network (one with one hidden layer) can implement any arbitrarily complex decision boundary (function) given a sufficient number of neurons in the hidden layer. Such a network was adopted for this research. The number of neurons in the hidden layer was determined empirically to be ten for the given two datasets.

The number of neurons in the input layer is commonly chosen to be equal to the dimensionality of the input feature vectors while the number of neurons in the output layer is always equal to the number of classes that the neural network has been trained to recognize. Following these approaches, for the current research, the number of neurons in the input layer was made to be one for the free flowing traffic dataset and two for the slow moving dataset. This is because the feature spaces became 1-dimensional and 2 – dimensional for the free flowing and slow moving traffic datasets respectively after the feature extraction process. Similarly, the number of neurons in the output layer was set to three for the two datasets because there were three vehicle classes: small, medium and large vehicle classes.

The activation function

In most classification problems, the sigmoid function is commonly used as the activation function for the hidden neurons while the soft-max function is commonly used for the output layer neurons. These activation functions were similarly used in the current design. The mathematical expressions of these two activation functions have been given in section 2.2.3.2.

Training the network: Error back propagation algorithm

The training of the multilayer neural network classifier was done by the error back propagation algorithm detailed in section 2.2.3.3. There are many variants of the error back-propagation algorithm. However, second order based techniques are generally faster than their first order counterparts and are therefore commonly used in many practical problems. Similarly, in this work, the scaled conjugate gradient back-propagation algorithm [28] [31] was used with the batch training protocol. The batch training protocol was used despite the fact that it is slower than the stochastic training protocol (the other most common training protocol) because it accommodates second order techniques which are much faster than the first order techniques [22].

For the training of this particular classifier in the current work, each training subset was subdivided into two: training and validation subsets. The validation subset validated the performance of the classifier after each iteration of training using the cross-entropy error as a measure of training performance. This helped to determine the best point to stop training the classifier and so obtain the best performance for generalization. The usage of the cross-entropy error measure function rather than the sum-of-squares error function (the other common measure of training error) for training a neural network in classification problems results in faster training and better generalization performance [69].

Given N training samples, and the desired and actual outputs of the neural network classifier as t_n and y_n respectively, the cross-entropy error function for the network can be expressed as [69]:

$$E = -\sum_{n=1}^{N}\{(t_n \ln y_n) + (1 - t_n)\ln(1 - y_n)\} \tag{3.21}$$

3.2.4 Designing the Ensemble classifier

The designed individual classifiers were combined in parallel to form an ensemble classifier so as to try and improve the classification accuracy. The base classifiers were dissimilar: the k – nearest neighbours classifier, the nearest–centroid classifier, the naïve Bayes classifier and the multilayer neural network classifier. However, “model combination is not a magical

formula always guaranteed to decrease error; base-learners should be diverse and accurate — that is, they should provide useful information" [20].

In combining the classifiers, two questions had to be addressed [20] [23]:

(i) How to generate base classifiers that complement each other, i.e. diverse

(ii) How to combine the outputs of the base classifiers for maximum accuracy

3.2.4.1 Generating diverse base classifiers

A number of approaches used to generate diverse base classifiers have been given in section 2.2.4.1. Other than using dissimilar base classifiers for the ensemble classifier, different training sets were used for each base-classifier during any of the 5-fold cross-validation trials. In the first trial, for example, of the five subsets, subset one was used for testing while subsets two, three, four, and five were used for training the k-nearest neighbours, the nearest centroid, the naïve Bayes and the multilayer neural network base classifiers respectively. During the second trial, subset two was used for testing while subsets, three, four, five and one were used for training the k-nearest neighbours, the nearest centroid, the naïve Bayes and the multilayer neural network base classifiers respectively. The same approach was used for the three remaining trials. Then, the overall classification accuracy of the ensemble classifier on a given dataset was obtained by averaging its classification accuracies for the five trials.

3.2.4.2 Combining the outputs of the base classifiers

Before combining the outputs of the base classifiers, the types of outputs [24] from all the base classifiers have to be the same. In the current work, the k-nearest neighbours, the nearest centroid and the naïve Bayes classifiers gave type one outputs while the multilayer neural network classifier gave a type three output. Thus, before the combination, the multilayer neural network classifier output (type three output) was converted into a type one output by selecting the class having the highest confidence rating as the output of the classifier. Hence, after the conversion, all the outputs of the base classifiers became type one and therefore ready for combination.

The weighted majority voting method was used to combine the outputs of the base classifiers due to its simplicity and intuitive attributes. Using this method, the output of the ensemble

classifier was a linear combination of the outputs of its base classifiers. The base classifiers were assigned different weights on the basis of their individual performance. The weight assigned to a given base classifier for a given trial was always equal to its normalized classification accuracy in that trial. This guaranteed 'fairness' in the assignment of the weights.

Given an unknown feature vector $\vec{x}$, the decision functions for each pattern class were computed using the assigned weights and then $\vec{x}$ was assigned to the class whose decision function yielded the largest numerical value. This can be generalized as follows [36]:

Let the number of base classifiers be C and the number of pattern classes be W. Let the value $d_{i,j}$ represent the support for a hypothesis that the unknown feature vector $\vec{x}$ to be classified comes from class w_j . The outputs of the base classifiers can be represented as degrees of support for the classes as:

$$d_{i,j} = \begin{cases} 1 & \text{if } C_i \text{ classifies } \vec{x} \text{ in } w_j \\ 0 & \text{otherwise} \end{cases} \tag{3.22}$$

The decision function for class w_j is computed as:

$$g_j(\vec{x}) = \sum_{i=1}^{C} b_i d_{i,j} \tag{3.23}$$

Where b_i is the normalized weight of classifier C_i. Thus given an unknown feature vector $\vec{x}$, assign it to class w_k if

$$g_k(\vec{x}) = \sum_{i=1}^{C} b_i d_{i,k} = \max_{i=1}^{W} \sum_{i=1}^{C} b_i d_{i,j} \tag{3.24}$$

In the current case, $C = 4$ and $W = 3$.

Figure 3.13 shows a simplified diagrammatic design of the ensemble classifier.

Many algorithms for improving the performance of the ensemble classifiers have been put forward. They include boosting, bagging and random subspace sampling [19] [23] [36]. However, boosting and bagging algorithms have been established to be only useful for

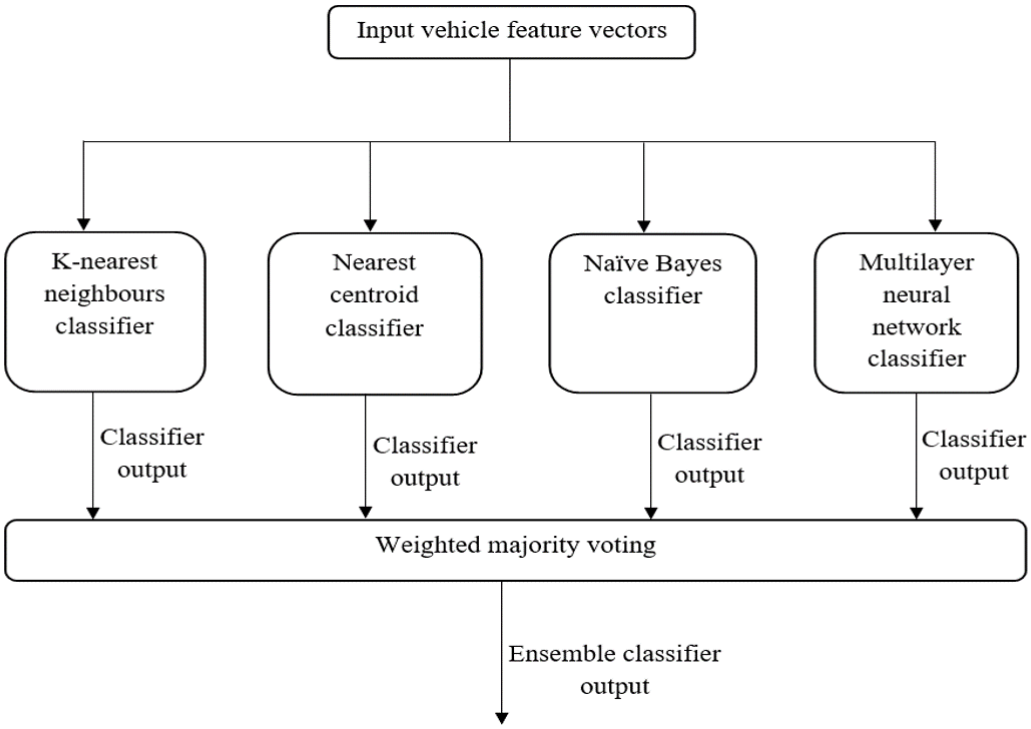


Figure 3.13: The Ensemble pattern classifier

unstable base classifiers such as neural networks and decision trees [19] [23]. They will not improve significantly the performance of the ensemble classifier in the current case since other than the neural network classifier, all the other base classifiers are stable; that is, small changes in their inputs will not cause large changes in the outputs. With that notwithstanding, bagging was directly catered for by use of different training sets for different base classifiers during any classification trial (section 3.2.4.1). This is a better approach which promotes diversity of the training set, and therefore the base classifiers, than the original bagging algorithm in which some samples are simultaneously used to train different base classifiers. Random subspace sampling is only applicable for multivariate feature spaces. But in this work, the feature space of the free flowing traffic dataset became 1-dimensional after the application of the PCA algorithm (section 3.2.2); therefore, a random subspace ensemble classifier is not applicable for this dataset.

Although it is theoretically possible to use the random subspace sampling technique for the slow moving traffic dataset (its feature space is now 2 – dimensional after the application of the PCA algorithm), practically, it led to deterioration of the ensemble classification accuracy by 5.9%. This is because the dataset's new two features contained 87.61% and 7.29% of information about the dataset (section 3.2.2), and as such, 7.29% of information about a

dataset alone is not sufficient to form a basis for classification on its own. Therefore, random subspace sampling technique was not used finally.

3.2.4.3 Pruning the Ensemble classifier

After the development of the ensemble classifier, redundant base classifiers were pruned off using the backward selection method. Using this method, one classifier was removed from the ensemble classifier at a time (with replacement) and then the classification accuracy of the remnant ensemble classifier was noted. The removed base classifier was subsequently replaced and then a different one was removed. The classification accuracy of the remnant ensemble classifier was again noted. This procedure was repeated for all the four base classifiers. The base classifier whose removal caused least deterioration of the classification accuracy of the ensemble classifier was pruned off. The pruning procedure continued until it could not be taken any further as explained in subsequent paragraphs. After the pruning, the classification accuracies of the remnant ensemble classifiers obtained at the various stages of pruning were compared to that of the original ensemble classifier and then the best among them was chosen as the new ensemble classifier. Hence, if none of the remnant ensemble classifiers were better than the original ensemble classifier, the original ensemble classifier was retained.

For purposes of the current discussion, the classification accuracies of the original (non-pruned) ensemble classifier and its four base classifiers (all obtained through 5 – fold cross-validation) on the two datasets are given in Table 3.1. Ground truth was determined using human visual inspection.

Table 3.1: Classification accuracies of the ensemble classifier and its base classifiers

Dataset	**Classification accuracy**				
	K-nearest neighbours classifier	Nearest centroid classifier	Naïve Bayes Classifier	Multilayer neural network classifier	Ensemble classifier
Free flowing traffic	90.9%	89.4%	91.1%	89.3%	91.4%
Slow moving traffic	80.4%	77.2%	75.9%	77.7%	82.2%

For the free flowing traffic dataset, the classification accuracy of the ensemble classifier with all the four base classifiers (before pruning) from Table 3.1 is 91.4%. In the first step of pruning, the base classifiers were removed from the ensemble classifier one at a time with replacement in the following order: the multilayer neural network classifier, the naïve Bayes classifier, the nearest centroid classifier and the k-nearest neighbours classifier. The classification accuracy of the remnant ensemble classifier after the removal of the base classifiers in the above mentioned order were noted to be: 90.6%, 91.4%, 91.1% and 91.0% respectively. Thus the removal of the naïve Bayes classifier from the ensemble classifier resulted in the least deterioration of the classification accuracy (by 91.4% - 91.4% = 0.0%) of the ensemble classifier and hence was pruned off. The remnant ensemble classifier now had three base classifiers: the k-nearest neighbours, the nearest centroid and the multilayer neural network classifiers. Once again, the pruning procedure was repeated for these remaining base classifiers. They were removed in the following order: the multilayer neural network classifier, the nearest centroid classifier and the k-nearest neighbours classifier. The classification accuracy of the ensemble classifier after each removal became: 91.4%, 91.8% and 91.1% respectively. The removal of the nearest centroid classifier resulted in the least deterioration of the ensemble classification accuracy (by 91.4% - 91.8% = - 0.4%). It was therefore pruned off leaving the remnant ensemble classifier with only two base classifiers: the k-nearest neighbours and the multilayer neural network classifiers. Note that this remnant ensemble classifier need not be pruned further because its two base classifiers had already been determined (Table 3.1) to have inferior classification accuracies to that of the original ensemble classifier. Since this remnant ensemble classifier (having two base classifiers) was not only more accurate than the remnant ensemble classifier of the previous pruning stage,

but also had better classification accuracy than the original ensemble classifier (91.8% vs. 91.4%), it was picked as the new ensemble classifier for the free flowing traffic dataset.

Similarly for the slow moving traffic dataset, the classification accuracy of the ensemble classifier with all the four base classifiers from Table 3.1 is 82.2%. In the first step of pruning, the base classifiers were removed from the ensemble classifier one at a time with replacement in the following order: the multilayer neural network classifier, the naïve Bayes classifier, the nearest centroid classifier and the k-nearest neighbours classifier. The classification accuracy of the remnant ensemble classifier after the removal of the base classifiers in the above mentioned order were noted to be: 78.1%, 79.9%, 79.9% and 80.8% respectively. Thus the removal of the k-nearest neighbours classifier from the ensemble classifier resulted in the least deterioration of the classification accuracy (by 82.2% - 80.8% = 1.4%) of the ensemble classifier and hence was pruned off. The remnant ensemble classifier now had three base classifiers: the nearest centroid, the naïve Bayes and the multilayer neural network classifiers. Once again the pruning procedure was repeated for these remaining base classifiers. They were removed in the following order: the multilayer neural network classifier, the naïve Bayes classifier and the nearest centroid classifier. The classification accuracy of the ensemble classifier after each removal became: 79.0%, 77.7% and 81.1% respectively. The removal of the nearest centroid classifier resulted in the least deterioration of the ensemble classification accuracy (by 82.2% - 81.1% = 1.1%). It was therefore pruned off leaving the ensemble classifier with only two base classifiers: the naïve Bayes and the multilayer neural network classifiers. Similar to the case of the free flowing traffic dataset, this remnant ensemble classifier need not be pruned further since its two base classifiers had already been determined (Table 3.1) to have inferior classification accuracy to that of the original ensemble classifier. However, it is noted that the two remnant ensemble classifiers have inferior classification accuracies (80.8% and 81.3%) to that of the original ensemble classifier (82.2%). Therefore, the original ensemble classifier was retained as the ensemble classifier for the slow moving traffic dataset.

Chapter Four

EXPERIMENTAL RESULTS AND DISCUSSION

The data for this research was acquired as explained in section 3.2.1. As detailed in that section, the data for the free flowing traffic dataset was collected at three different time periods of the day: morning, midday and evening hours. However, a single dataset was collected for the slow moving traffic and the time period or the existing illumination condition at the time of the data collection was not a factor as explained in the referred section. In all the results obtained in this research, ground truth was determined by human visual inspection.

4.1 Vehicle detection results

For the free flowing traffic dataset, the morning dataset had 220 vehicles in the video data, 209 of which were correctly detected. This translates to 95% detection accuracy. In order to obtain as many vehicles as possible for classification, temporary manual adjustments were done to the vehicle detection algorithm for the frames whose vehicles were not correctly detected and as a result, 216 of the 220 vehicles were extracted for classification, while the other 4 were over-segmented and were therefore not included in the classification. The same was done for the other two datasets from the other two different time periods. The vehicle feature vectors extracted from the three time periods of the day were combined to form an overall dataset of the free flowing traffic dataset for classification.

For the slow moving traffic dataset, there were a total of 246 vehicles in the video data, 202 of which were correctly detected. This translates to 82.1% detection accuracy. After temporary manual manipulations on the frames whose vehicles were not correctly detected as explained for the free flowing traffic dataset, 224 vehicles were extracted for classification.

The vehicle detection results for both free flowing traffic and slow moving traffic datasets are given in Table 4.1.

Table 4.1: A summary of the detection results

Dataset	Total no. of vehicles in the video data	Total no. of correct detections	Total no. of wrong detections	Detection accuracy
Morning	220	209	11	95.0%
Midday	306	291	15	95.1%
Evening	438	425	13	97.0%
Overall(Free flowing traffic)	964	925	39	96.0%
Slow moving traffic	246	202	42	82.1%

The vehicle detection accuracy for the slow moving traffic dataset was generally poorer than those for the free flowing traffic datasets (Table 4.1). The reason for this is that occlusions were more severe in the slow moving traffic scene than for the free flowing traffic scenes. Consequently, at times, two or even more vehicles could be detected as one vehicle rather than as separate vehicles (Figure 3.12).

4.2 Vehicle classification results

There were two datasets for classification: the free flowing traffic dataset and the slow moving traffic dataset. The classification on both datasets was carried out through cross-validation. The cross-validation technique was carried out for all the classifiers under consideration. The folds for cross-validation were fixed to ensure that all the individual classifiers and the ensemble classifier were trained using the same feature vectors, and then tested using the same feature vectors during any trial. The fixed folds not only made it possible to combine the outputs of the base classifiers, but also made comparisons among the classification accuracies fair and realistic, rather than the random non-fixed folds in which, for example, the test folds are not guaranteed to be the same for all classifiers.

The classification algorithms under consideration were: the k – nearest neighbours, the nearest centroid, the naïve Bayes, the multilayer neural network and the ensemble classifiers.

4.2.1 Free flowing traffic dataset

For this dataset, the ensemble classifier had two base classifiers: the k-nearest neighbours and the multilayer neural network classifiers (section 3.2.4.3). Consequently, for convenience (since the folds were fixed), 3–fold cross-validation technique was used for classification on the free flowing traffic dataset by the ensemble classifier. However, for the individual classification algorithms, 5-fold cross-validation technique was used as it was empirically determined to be sufficient.

Therefore, for the ensemble classifier, the dataset was broken down into three subsets for 3-fold cross-validation. In the first trial, subset one was used for testing while subset two was used to train the k – nearest neighbours classifier and subset three was used for training the multilayer neural network classifier. In the second trial, subset two was used for testing while subset three was used for training the k – nearest neighbours classifier and subset one was used for training the multilayer neural network classifier. In the third and last trial, subset three was used for testing while subset one was used for training the k –nearest neighbours classifier and subset two was used for training the multilayer neural network classifier. Training the base classifiers in this way ensures diversity (section 3.2.4.1). The classification accuracy of the ensemble classifier was obtained by averaging the classification accuracies of the three trials.

Regarding the individual classification algorithms in which 5-fold cross-validation method was used, one subset was used for testing while the other four were used for training the algorithm during any trial. To guarantee stable outputs by the multilayer neural network and the ensemble classifiers, the starting point of the MATLAB random number generator was fixed. This guaranteed that the initial weights assigned to the multilayer neural network were fixed at any time.

Table 4.2 gives a summary of the vehicle classification results by the five classifier algorithms on the free flowing traffic dataset.

Table 4.2: A summary of classification results on the free flowing traffic dataset

Total no. of vehicles	**No. of small vehicles**	**No. of medium vehicles**	**No. of large vehicles**	**Classification accuracy**				
				K-nearest neighbor classifier	Nearest centroid classifier	Naïve Bayes Classifier	Multilayer neural network classifier	Ensemble classifier
952	594	199	159	90.9%	89.4%	91.1%	89.3%	91.8%

As can be seen from Table 4.2, the ensemble classifier had the best classification accuracy. Its two base classifiers (the k – nearest neighbours and the multilayer neural network) complemented each other to achieve a classification accuracy of 91.8% compared to the best individual classifier (the naïve Bayes) which had a classification accuracy of 91.1%.

The classification accuracies of the ensemble classifier on each of the three test subsets of the free flowing traffic dataset are given by the confusion matrices in Tables 4.3 – 4.5.

Table 4.3: Test subset 1

		Obtained class labels		
		Small vehicle	Medium vehicle	Large vehicle
True class labels	Small vehicle	229	9	0
	Medium vehicle	16	29	3
	Large vehicle	0	2	29
Classification accuracy		90.5%		

Table 4.4: Test subset 2

		Obtained class labels		
		Small vehicle	Medium vehicle	Large vehicle
True class labels	Small vehicle	182	2	0
	Medium vehicle	15	54	1
	Large vehicle	1	5	57
Classification accuracy			92.4%	

Table 4.5: Test subset 3

		Obtained class labels		
		Small vehicle	Medium vehicle	Large vehicle
True class labels	Small vehicle	170	3	0
	Medium vehicle	17	63	1
	Large vehicle	1	2	62
Classification accuracy			92.5%	

The leading diagonal elements of a confusion matrix represent correctly classified vehicles while the off-diagonal elements represent misclassifications. Therefore, for a given confusion matrix, the classification accuracy is computed as:

$$\text{Classification accuracy} = \frac{\text{Sum of leading diagonal elements}}{\text{Sum of all matrix elements}} * 100 \tag{4.1}$$

The overall ensemble classification accuracy is obtained as:

Average classification accuracy = mean (90.5%, 92.4%, 92.5%) = 91.8%

4.2.2 Slow moving traffic dataset

The ensemble classifier for the slow moving traffic dataset has four base classifiers (section 3.2.4.3). Both the ensemble and the individual classification algorithms made use of 5-fold

cross-validation technique for reasons similar to those given in section 4.2.1. Similarly, the cross-validation technique was used by following the same steps as those explained in sections 3.2.3 and 4.2.1.

Table 4.6 gives a summary of the vehicle classification results by the five classifier algorithms on the slow moving traffic dataset.

Table 4.6: A summary of classification results on the slow moving traffic dataset

Total no. of vehicles	**No. of small vehicles**	**No. of medium vehicles**	**No. of large vehicles**	**Classification accuracy**				
				K-nearest neighbor classifier	Nearest centroid classifier	Naïve Bayes Classifier	Multilayer neural network classifier	Ensemble classifier
224	114	73	37	80.4%	77.2%	75.9%	77.7%	82.2%

Similar to the classification results obtained for the free flowing traffic dataset, for the slow moving traffic dataset, the ensemble classifier classification results were better than those obtained by the individual classification algorithms.

The classification results for the slow moving traffic dataset (Table 4.6) were generally poorer than those for the free flowing traffic dataset (Table 4.2). This is evident when the classification accuracies of the corresponding classification algorithms on the two datasets are contrasted. The reason for this is that occlusions were more severe in the slow moving traffic scene than for the free flowing traffic scenes. Consequently, at times, two or even more vehicles could be detected as one vehicle rather than as separate vehicles (Figure 3.12). This ultimately led to not only erroneous vehicle counts, but also in misclassifications.

Looking at the classification accuracies of the individual classification algorithms on the two datasets, it is of interest to note that for the free flowing traffic dataset, the naïve Bayes was the most accurate (91.1%) while the k – nearest neighbours was the most accurate (80.4%) on the slow moving traffic dataset. These results confirm the 'No Free Lunch' theorems which say that there is no single classification algorithm that is always guaranteed to be the best under every circumstance. Therefore, it is not possible to know in advance which classification algorithm will be the best for a given task. The classification accuracies of

the ensemble classifier on each of the five test subsets of the slow moving traffic dataset are given by the confusion matrices in Tables 4.7 – 4.11.

Table 4.7: Test subset 1

		Obtained class labels		
		Small vehicle	Medium vehicle	Large vehicle
True class labels	Small vehicle	21	2	0
	Medium vehicle	1	13	0
	Large vehicle	0	2	5
Classification accuracy			88.6%	

Table 4.8: Test subset 2

		Obtained class labels		
		Small vehicle	Medium vehicle	Large vehicle
True class labels	Small vehicle	17	6	0
	Medium vehicle	1	14	0
	Large vehicle	0	2	6
Classification accuracy			80.4%	

Table 4.9: Test subset 3

		Obtained class labels		
		Small vehicle	Medium vehicle	Large vehicle
True class labels	Small vehicle	20	3	0
	Medium vehicle	3	11	0
	Large vehicle	0	2	5
Classification accuracy			81.8%	

Table 4.10: Test subset 4

		Obtained class labels		
		Small vehicle	Medium vehicle	Large vehicle
True class labels	Small vehicle	18	5	0
	Medium vehicle	2	11	2
	Large vehicle	0	0	8
Classification accuracy			80.4%	

Table 4.11: Test subset 5

		Obtained class labels		
		Small vehicle	Medium vehicle	Large vehicle
True class labels	Small vehicle	16	6	0
	Medium vehicle	1	14	0
	Large vehicle	1	1	5
Classification accuracy			79.5%	

Average classification accuracy = mean (88.6%, 80.4%, 81.8%, 80.4%, 79.5%) = 82.2%.

For both free flowing and slow moving traffic datasets, the relatively low camera position was the main cause of detection errors since it was difficult to 'see' the spaces between the vehicles on the same lane when the vehicles involved were very close together as seen in Fig. 3.12. It was also the main cause of misclassification since errors in vehicle detection led to noisy data for classification. Besides the noisy data for classification, the relatively low camera position meant that, for small vehicles, for example, due to their size, their entire tops were visible while this was not true for the other classes where only the fronts and some part of the tops were visible; resulting in the usage of different dimensions for different vehicles in classification. This greatly reduced the ability of the classifier to distinguish between small

and medium vehicles as is evident from the given confusion matrices. From the confusion matrices, it is of interest to note that misclassifications between small and large vehicles were rare. This was due to the fact that the visible parts of large vehicles were generally larger than the visible parts of most small vehicles, making it easier for the classifier to distinguish between the two classes. Attempts to raise the camera position to be high enough relative to the Region Of Interest (ROI) were not successful due to practical limitations.

The above presented results are comparable to those published in the literature [41] [42] [44] – [48] [50] [52] [53] [58] [59] [60] [62]; where on average, average detection and classification accuracies of between 80% - 100% are reported. However, it should be noted that a one to one comparison between the performances of any two systems is only realistic if, among others, the same dataset was used to test them [19] [22] [23]. Such a dataset(s) could not be obtained for this research.

Chapter Five

CONCLUSION

The main goal of this research was to develop a vision based vehicle detection and classification system that will be useful not only for free flowing traffic scenes in which background modelling and subtraction alone is sufficient for vehicle detection, but also for slow moving or stationary traffic scenes in which meaningful and robust backgrounds are unattainable. The system was meant to work reasonably well under different illumination conditions during the day and be able to reliably handle shadows. All these were to be done using traffic surveillance videos collected from natural traffic scenes. Regarding vehicle classification, an ensemble classifier was to be developed using the most common classification algorithms and then its classification accuracy on the given datasets compared to those of its base classifiers. Using the comparisons, recommendations were to be made on which way to go for traffic management systems.

The proposed algorithm detected vehicles with a good degree of accuracy under different illumination conditions during the day for both free flowing and slow moving traffic scenes, with relatively good computational efficiency, taking 5.4 seconds per frame on MATLAB R2015a, using an Intel(R) Core(TM) i3-3110M CPU @ 2.40GHz 2.40GHz Laptop computer. This computational speed can significantly improve if dedicated software and hardware were used. The shadows were also well handled with a good degree of success as illustrated in Figure 3.5. The vehicle detection algorithm used a novel approach in which the vehicles were simultaneously extracted from the traffic video data frames and their negatives using the Laplacian of a Gaussian edge detector. Edge linking was achieved through mathematical morphology and summation of the positive and negative edge maps. However, despite the success of the algorithm, it was noted that over-segmentation occurred for very large trucks: with cabins and their trailers being detected as separate vehicles. The algorithm also had problems with occluded vehicles as seen in Figure 3.12. Nevertheless, the algorithm gave promising results for the slow moving traffic scene where occlusions

were more severe. To minimize occlusion problems, it is suggested to raise the position of the camera to be high enough with respect to the ROI. This was not practical in the current research.

In this research work, a comparison of the most common classification algorithms used in classifying vehicles and their ensemble was made. These were: the k-nearest neighbours, the nearest centroid, the naïve Bayes and the multilayer neural network classifiers. The objective of the comparisons was to identify the 'best' classification algorithm out of the classifiers under consideration for this kind of task. However, as demonstrated by the presented classification results and as explained in section 4.2.2, there is no single classifier that is clearly the best for every dataset. However, in general, the classification accuracies of the ensemble classifier on both datasets was slightly better than that of any of its base classifiers. Nevertheless, the ensemble classifier was much more complex computationally than any of the individual classification algorithms. This may make it unsuitable for real time systems. Therefore, rather than the use of an ensemble classifier, for real time vision based traffic management systems, this research concludes that in a given scenario, several classification algorithms be tried out on small datasets and then the best one among them be chosen for deployment. In this way, relatively good classification results can be obtained at a lesser computational cost than if an ensemble classifier were used. For example, from Table 4.2, the classification accuracy of the ensemble classifier on the free flowing traffic dataset is given as 91.8% while that of the naïve Bayes classifier is 91.1%. Therefore, in this case, this research work recommends the adoption of the naïve Bayes classifier for real time deployment rather than the ensemble classifier. This is because the naïve Bayes classifier was simpler in design (it involved implementing only a single decision function) and was computationally more efficient than the ensemble classifier, taking only 0.6 seconds to execute compared to the 27.8 seconds it took to execute the ensemble classifier on the same dataset, using the same computer. With that notwithstanding, for non-real time traffic management systems, ensemble classifiers will be quite useful since for such systems, accuracy is the most important thing.

The results obtained by the proposed system are comparable to those published in the literature [41] [42] [44] – [48] [50] [52] [53] [58] [59] [60] [62]; where on average, average

detection and classification accuracies of between 80% - 100% are reported. However, it should be noted that a one to one comparison between the performances of any two systems is only realistic if, among others, the same dataset was used to test them [19] [22] [23]. Such a dataset(s) could not be obtained for this research.

Chapter Six

RECOMMENDED FURTHER WORK

This work could be extended to incorporate the hardware specifications of the cameras into the vehicle detection algorithm. For example, it was noted after trying different cameras that the camera resolution has an implicit relationship with the sizes of the required structuring elements and the appropriate size of the region of interest. In this respect, it was noted that the higher the resolution of the camera, the bigger the size of the structuring elements required. This can be investigated further so as to come up with recommendations for real time applications. This need was also identified in [55]. In addition to this, an approach for determining optimal sizes of structuring elements with respect to inter-vehicle distances need to be developed. Other weather conditions such as rainy condition could also be investigated and the algorithm developed in such a way that it is able to adapt to such conditions. Besides these, night detection and occlusions in traffic scenes which have remained a major problem for vision based traffic management systems for many years need to be solved.

On vehicle classification, more classification algorithms such as the support vector machines, decision trees, and clustering methods would be investigated and incorporated in the comparisons.

Finally, the proposed system need to be deployed by incorporating it into existing road traffic monitoring systems and integrated with mobile telephony systems so that road users can be able to know the traffic situation on a given road section at any time of the day.

REFERENCES

[1] S. Sivaraman and M. M. Trivedi, ”A Review of recent developments in vision-based vehicle detection,” in 2013 IEEE Intelligent Vehicles Symposium (IV), Gold Coast, Australia, 2013.

[2] K. Kim, T. H. Chalidabhongse, D. Harwood and L. Davis, ”Real-time foreground–background segmentation using codebook model,” Elsevier, 2005.

[3] C. Stauffer and W. E. Grimson, ”Adaptive background mixture models for real-time tracking,” in IEEE International Conference on Computer Vision and Pattern Recognition, 1999.

[4] R. Maini and H. Aggarwal, ”A Comprehensive Review of Image Enhancement Techniques,” Journal of Computing, vol. 2, no. 3, pp. 8-13, 2010.

[5] R. C. Gonzalez and R. E. Woods, Digital image processing, Third Edition, Upper Saddle River, New Jersey: Pearson Education Inc., 2008.

[6] A. C. Bovik, The Essential Guide to Image Processing, San Diego, California, USA: Elsevier Inc., 2009.

[7] W. K. Pratt, Digital Image Processing, Third Edition, New York: John Wiley Sons, Inc., 2001.

[8] F. Y. Shih, Image Processing and Pattern Recognition, Fundamentals and Techniques, Hoboken, New Jersey: John Wiley Sons, Inc., 2010.

[9] J. C. Russ, The Image Processing Handbook, Sixth Edition, Boca Raton, USA: Taylor Francis Group, 2011.

[10] The MathWorks, Inc., ”Image Processing Toolbox, User’s Guide,” The MathWorks, Inc., Natick, USA, 2015.

[11] A. Raju, G. S. Dwarakish and D. V. Reddy, ”A Comparative Analysis of Histogram Equalization based Techniques for Contrast Enhancement and Brightness Preserving,” International Journal of Signal Processing, Image Processing and Pattern Recognition, vol. 6, no. 5, pp. 353-366, 2013.

[12] H. K. Sawant and M. Deore, ”A Comprehensive Review of Image Enhancement Techniques,” International Journal of Computer Technology and Electronics Engineering (IJCTEE), vol. 1, no. 2, pp. 39-44, 2011.

[13] S. Nimkar, S. Shrivastava and S. Vargh, ”Contrast Enhancement and Brightness Preservation using Multi-Decomposition Histogram Equalization,” Signal Image Processing : An International Journal (SIPIJ), vol. 4, no. 3, pp. 83-93, 2013.

[14] P. K. Garg, P. Verma and A. Bhardwaz, ”A Survey Paper on Various Median Filtering Techniques for Noise Removal from Digital Images,” American International Journal of Research in Formal, Applied Natural Sciences, vol. 7, no. 1, pp. 43-47, 2014.

[15] R. C. Gonzalez, R. E. Woods and S. L. Eddins, Digital Image Processing using MATLAB, Second Edition, USA: Gatesmark, LLC, 2009.

[16] T. S. Huang, G. J. Yang and G. Y. Tang, ”A Fast Two-Dimensional Median Filtering Algorithm,” IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 27, no. 1, pp. 13-18, 1979.

[17] T. Acharya and A. K. Ray, Image Processing, Principles and Applications, Hoboken, New Jersey. John Wiley Sons, Inc., 2005.

[18] G. X. Ritter and J. N. Wilson, Handbook of Computer Vision Algorithms in Image Algebra, Second Edition, Boca Raton, Florida: CRC Press LLC, 2001.

[19] G. Dougherty, Pattern Recognition and Classification, An Introduction, New York: Springer Science+Business Media, 2013.

[20] E. Alpaydın, Introduction to Machine Learning, Second Edition, Massachusetts: Massachusetts Institute of Technology, 2010.

[21] K. Fukunaga, Introduction to Statistical Pattern Recognition, Second Edition, San Francisco: Morgan Kaufmann, 1990.

[22] R. O. Duda, P. E. Hart and D. G. Stork, Pattern Classification, Second Edition, John Wiley Sons, Inc., 2000.

[23] A. R. Webb and K. D. Copsey, Statistical pattern recognition, Third Edition, Chichester, West Sussex, UK: John Wiley Sons, Ltd, 2011.

[24] J. R. Parker, Algorithms for Image Processing and Computer Vision, Second Edition, Indianapolis, USA: Wiley Publishing, Inc., 2011.

[25] The MathWorks, Inc., "Statistics and Machine Learning Toolbox, User's Guide," The MathWorks, Inc., Natick, USA, 2015.

[26] C. M. Bishop, Pattern Recognition and Machine Learning, Singapore: Springer Science+Business Media, LLC, 2006.

[27] Scholastic Video Book Series, Artificial Neural Networks, Part 1, 2013.

[28] The MathWorks, Inc., "Neural Network Toolbox, User's Guide," The MathWorks, Inc., Natick, USA, 2015.

[29] P. Picton, Introduction to Neural Networks, London: Macmillan Press Ltd, 1994.

[30] L. Prevost, S. Marinai and F. Schwenker, Artificial Neural Networks in Pattern Recognition, Berlin, German: Springer Science+Business Media, 2008.

[31] C. M. Bishop, Neural Networks for Pattern Recognition, Oxford: Clarendon Press, 1995.

[32] D. Graupe, Principles of Artificial Neural Networks, Second Edition, Danvers, USA: World Scientific Publishing Co. Pte. Ltd., 2007.

[33] J. Kittler, M. Hatef, R. P. Duin and J. Matas, "On Combining Classifiers," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 20, no. 3, pp. 226-239, 1998.

[34] Q. Liu, J. Lu, S. Chen and K. Zhao, ”Multiple Naïve Bayes Classifiers Ensemble for Traffic Incident Detection,” Mathematical Problems in Engineering, Hindawi Publishing Corporation, vol. 2014, 2014.

[35] J. Kittler, ”Combining Classifiers: A Theoretical Framework,” Pattern Analysis Applications, pp. 18-27, 1998.

[36] L. I. Kuncheva, Combining Pattern Classifiers, Methods and Algorithms, Hoboken, New Jersey. John Wiley Sons, Inc., 2004.

[37] V. Heidari and M. R. Ahmadzadeh, ”A method for vehicle classification and resolving vehicle occlusion in traffic images,” Pattern Recognition and Image Analysis (PRIA), 2013 First Iranian Conference on, pp. 1-6, 6-8 March 2013.

[38] C. C. C. Pang, W. W. L. Lam and N. H. C. Yung, ”A Novel method for resolving vehicle occlusion in a monocular traffic-image sequence,” IEEE Transactions on Intelligent Transportation Systems, vol. 5, no. 3, pp. 129-141, 2004.

[39] A. Sharma, D. Bullock, S. Peeta and J. Krogmeier, ”Detection of inclement weather conditions at a signalized intersection using a video image processing algorithm,” The Civil Engineering at DigitalCommons@University of Nebraska - Lincoln, vol. 32, pp. 150-155, 24-27 September 2006.

[40] Minnesota Department of Transportation, Office of Traffic, Safety Technology and SRF Consulting Group, Inc., ”Evaluation of non-intrusive technologies for traffic detection,” United States Department of Transportation, Federal Highway Administration, Minneapolis, 2010.

[41] S. H. Yu, J. W. Hsieh, Y. S. Chen and W. F. Hu, ”An automatic traffic surveillance system for vehicle tracking and classification,” in Scandinavian Conference on Image Analysis (SCIA2003), Göteborg, Sweden, 2003.

[42] R. P. Avery, Y. Wang and G. S. Rutherford, ”Length-based vehicle classification using images from uncalibrated video cameras,” University of Washington, 22 July 2004. [Online]. Available:

http://faculty.washington.edu/yinhai/wangpublication_files/ITSC_05_VC. [Accessed 21 March 2015].

[43] Open Network Video Interface Forum Inc., "ONVIF™ core specification," August 2013. [Online]. Available: http://www.onvif.org/specs/core/ONVIF-Core-Specification-v240.pdf. [Accessed 20 June 2015].

[44] A. Ambardekar, M. Nicolescu, G. Bebis and M. Nicolescu, "Vehicle classification framework: a comparative study," EURASIP Journal on Image and Video Processing, 2014, **2014**:29.

[45] H. Rabiu, "Vehicle detection and classification for cluttered urban intersection," International Journal of Computer Science, Engineering and Applications (IJCSEA), vol. 3, no. 1, pp. 37-47, 2013.

[46] A. R. M. Ribeiro and P. L. Correia, "Automatic vehicle counting based on surveillance video streams," Instituto de Telecomunicações, Lisbon, Portugal, 2012.

[47] N. Buch, J. Orwell and S. A. Velastin, "Detection and classification of vehicles for urban traffic scenes," in Visual Information Engineering, 2008. VIE 2008. 5th International Conference, Xian China, 2008.

[48] P. K. Mishra, M. Athiq, A. Nandoriya and S. Chaudhuri, "Video-based vehicle detection and classification in heterogeneous traffic conditions using a novel kernel classifier," IETE Journal of Research, vol. 59, no. 5, pp. 541-550, 2013.

[49] I. Vujović, M. Jurčević and I. Kuzmanić, "Traffic video surveillance in different weather conditions," Transactions on Maritime Science, vol. 3, no. 1, pp. 32-41, 2014.

[50] V. Kwigizile, M. Selekwa and R. Mussa, "Highway vehicle classification by probabilistic neural networks," [Online]. Available: https://www.aaai.org/Papers/FLAIRS/2004/Flairs04-114.pdf. [Accessed 10 May 2015].

[51] Cambridge Systematics, Inc., ”FHWA Vehicle Classes With Definitions,” California Department of Transportation, [Online]. Available: https://www.dot.ca.gov/hq/tpp/offices/ogm/trucks/FHWA_Vehicle-Classes-With-Definitions%20cs. [Accessed 13 July 2015].

[52] O. Sidla, L. Paletta, Y. Lypetskyy and C. Janner, ”Vehicle recognition for highway lane survey,” in 7th International IEEE Conference on Intelligent Transportation Systems, 2004.

[53] M. Pancharatnam and D. U. J. Sonnadara, ”Vehicle counting and classification from a traffic scene,” in Proceedings of the 26th National IT Conference, Colombo, Sri Lanka.

[54] G. Zhang, R. P. Avery and Y. Wang, ”A Video-based vehicle detection and classification system for real-time traffic data collection using uncalibrated video cameras,” Department of Civil and Environmental Engineering, University of Washington, Washington, 2006.

[55] E. Ince, ”Measuring traffic flow and classifying vehicle types: a surveillance video based approach,” Turkish Journal of Electrical Engineering and Computer Science, vol. 19, no. 4, 2011.

[56] C. Ozkurt and F. Camci, ”Automatic Density Estimation and Vehicle Classification for Traffic Surveillance Systems using Neural Networks,” Mathematical and Computational Applications, vol. 14, no. 3, pp. 187-196, 2009.

[57] V. Jain, A. Sharma, A. Dhananjay and L. Subramanian, ”Traffic Density Estimation for Noisy Camera Sources,” in Transportation Research Board 91st Annual Meeting, Washington DC, 2012.

[58] J. Y. Ng and Y. H. Tay, ”Image-based Vehicle Classification System,” in The 11th Asia-Pacific ITS Forum Exhibition, Kaohsiung, Taiwan, 2011.

[59] B. Zhang and C. Zhao, ”Classification of Vehicle Make by Combined Features and Random Subspace Ensemble,” in 2011 Sixth International Conference on Image and Graphics, Hefei, Anhui, China, 2011.

[60] Y. Tang, Y. C. Xu and C. Z. Zhang, ”Robust Vehicle Detection Based on Cascade Classifier in Traffic Surveillance System,” The Open Automation and Control Systems Journal, vol. 6, pp. 349-354, 2014.

[61] A. Makris, M. Perrollaz, I. Paromtchik and C. Laugier, ”Integration of visual and depth information for vehicle detection,” Institut National de Recherche en Informatique et en Automatique (INRIA), Rhône-Alpes, 2011.

[62] T. Kowsari, S. S. Beauchemin and J. Cho, ”Real-time Vehicle Detection and Tracking Using Stereo Vision and Multi-View AdaBoost,” in 14th International IEEE Conference on Intelligent Transportation Systems, Washington, DC, USA, 2011.

[63] S. Abe, Support Vector Machines for Pattern Classification, Second Edition, London: Springer-Verlag London Limited, 2010.

[64] M. J. Zaki and W. Meira, Data Mining and Analysis, Fundamental Concepts and Algorithms, New York: Cambridge University Press, 2014.

[65] J. Han and M. Kamber, Data Mining: Concepts and Techniques, Second Edition, San Francisco: Elsevier Inc., 2006.

[66] D. Hand, H. Mannila and P. Smyth, Principles of Data Mining, Massachusetts: The MIT Press, 2001.

[67] S. A. Dudani, ”The distance-weighted k-nearest-neighbour rule,” IEEE Transactions on Systems, Man and, vol. 6, no. 4, p. 325–327, 1976.

[68] J. H. Friedman, J. L. Bentley and R. A. Finkel, ”An algorithm for finding best matches in logarithmic,” ACM Transactions on Mathematical Software, vol. 3, no. 3, pp. 209-226, 1977.

[69] P. Y. Simard, D. Steinkraus and J. C. Platt, ”Best practice for convolutional neural networks applied to visual document analysis.,” in Proceedings International Conference on Document Analysis and Recognition (ICDAR), 2003.

APPENDIX 1: MATLAB PSEUDO - CODE FOR VEHICLE DETECTION AND FEATURE EXTRACTION.

Vehicle detection and feature extraction

```
clear all % clear the workspace
clc % clear the command window
close all % close all figures except those on imtool
imtool close all % close all figures on imtool
```

Conversion of video to frames / frame extraction

```
videosource = VideoReader('Videofile'); % read the video file
vid = (videosource);
numFrames = vid.NumberOfFrames; % obtain the number of frames
n=numFrames;
 for i = 1:n
frames = read(vid,i);
 imwrite(frames,['Image' int2str(i), '.jpg']);% write the frames into disk
 im(i)=image(frames);

 end
```

Frame extraction & RGB to grayscale conversion

```
% load data to the workspace
Colour_frame = imread('Framefile');% read the frame data into the workspace
Gray_frame = rgb2gray(Colour_frame);% convert colour frame to gray frame

% display the frames and give them titles
figure,imshow(Colour_frame)
title('Original Colour Frame')
figure,imshow(Gray_frame)
title('Original Gray Frame')
```

Non-uniform illumination compensation using top-hat transformation

```
SE = strel('rectangle',ceil(size(Gray_frame)*1)); % structuring element
Background_compensated_frame = imtophat(Gray_frame,SE);% top-hat...
                                                       %transformation
Foreground_frame = Background_compensated_frame;
figure,imshow(Foreground_frame)% display the figure
title('Foreground Frame')
```

Image smoothing by median filtering of foreground frame

```
Smooth_frame = medfilt2(Foreground_frame,[3 3],'symmetric');...
                                % median filtering of the frame
figure,imshow(Smooth_frame)% display the figure
title('Median filtered Foregound Newframe ')
```

Aggressive blurring using Gaussian filter

```
Foregound_Newframe = imgaussfilt(Smooth_frame,5); % aggressive blurring ...
                                        %by Gaussian filter
figure, imshow(Foregound_Newframe)% display the figure
title('Gaussian Blurred Frame')
```

Contrast adjustment by contrast stretching

```
Foregound_Newframe = imadjust(Foregound_Newframe);

figure, imshow(Foregound_Newframe)% display the figure
title('Contrast Enhanced Image')
```

Edge detection and segmentation

```
Edge_map = edge(Foregound_Newframe,'log');% use LoG edge detector
figure, imshow(Edge_map);
title(' Positive Edge Map')

% SEGMENTATION
Binary_frame = Edge_map;
```

Region of interest modelling / ROI modelling / mask modelling

```
Region_interest_frame =  Colour_frame;

roi = Region_interest_frame;

roipoly(roi)% model the ROI polygon
c = [colun cordinates of the polygon];
r = [row cordinates of the polygon];

roi_mask = roipoly(roi,c,r);
figure,imshow(roi_mask)
title('ROI mask')
```

Negative transformation

```
Negative_Gray_frame = imcomplement(Gray_frame);% compute the negative frame
```

```
figure,imshow(Negative_Gray_frame);% display the figure
title('Negative Gray Frame')
```

Non-uniform illumination compensation using top-hat transformation

```
Background_compensated_frame = imtophat(Negative_Gray_frame,SE);% top-hat...
                        %transformation
Negative_Foreground_frame = Background_compensated_frame;
figure,imshow(Negative_Foreground_frame)% display the figure
title('Negative Foreground Frame')
```

Image smoothing by median filtering of foreground frame

```
Negative_Foreground_frame =medfilt2(Negative_Gray_frame,[3 3],'symmetric');
                % median filtering the frame
figure,imshow(Negative_Foregound_Newframe)% display the figure
title('Median filtered Negative Foreground Newframe ')
Negative_Foregound_Newframe = imgaussfilt(Negative_Gray_frame,3);
                % blurring the frame
figure,imshow(Negative_Foregound_Newframe)% display the figure
title('Blurred Negative Foreground Newframe ')
```

Contrast adjustment by contrast stretching

```
Negative_Foreground_frame = imadjust(Negative_Foregound_Newframe);

figure, imshow(Negative_Foreground_frame)% display the figure
title('Contrast Enhanced Negative Image')
```

Edge detection and segmentation

```
Negative_Edge_map = edge(Negative_Foreground_frame,'log');% use LoG edge...
                                                    %detector
figure, imshow(Negative_Edge_map)% display the figure
title('Negative Edge Map')
% SEGMENTATION
Negative_Binary_frame = Negative_Edge_map;
```

Adding positive and negative frames

```
Total_Edge_Map = imadd(Edge_map,Negative_Edge_map);

figure,imshow(Total_Edge_Map)% display the figure
title('Total Edge Map')
```

Image closing and filling

```
SE1 = strel('square',11);
Closed_image = imclose(Total_Edge_Map,SE1);
figure,imshow(Closed_image)% display the figure
title('Closeed Binary Image')
Filled_image = imfill(Closed_image,'holes');
figure,imshow(Filled_image)% display the figure
title('Filled Binary Image')
```

Skeletonizing

```
 Skeletonized_image = bwmorph(Filled_image,'skel',1);
 figure,imshow(Skeletonized_image)% display the figure
 title('Skeleton Image')
```

Pruning

```
prunned_image = bwmorph(Skeletonized_image,'spur',inf);
figure,imshow(prunned_image)% display the figure
title('Prunned Image')
```

Frame masking

```
%multiply the mask and the foreground frame
ROI_frame = immultiply(prunned_image,roi_mask);
figure,imshow(ROI_frame)% display the figure
title('ROI Frame')
```

Image opening

```
Opened_bw = bwareaopen(ROI_frame,500,8); % morphological opening of...
                                           %binary image
figure,imshow(Opened_bw);% display the figure
title('Opened Binary ROI Image')
```

Object labelling

```
L = bwlabel(Opened_bw);

s = regionprops(L, 'Centroid');
figure,imshow(Opened_bw)
title('Labeled Objects')
hold on
for k = 1:numel(s)
    c = s(k).Centroid;
    text(c(1), c(2), sprintf('%d', k), ...
        'HorizontalAlignment', 'center', ...
```

```
        'VerticalAlignment', 'middle');
end
hold off
```

Feature extraction

```
cc = bwconncomp(Opened_bw, 8) % connected components binary image
Number_objects = cc.NumObjects % number of connected components

for i = 1:Number_objects

imagecomponents = regionprops(cc,'Area','MajorAxisLength','MinorAxisLength',
'BoundingBox')
Areas = imagecomponents(i).Area
MajorAxisLengths = imagecomponents(i).MajorAxisLength
MinorAxisLengths = imagecomponents(i).MinorAxisLength

BoundingBoxes = imagecomponents(i).BoundingBox

BoundingBoxes_Areas(i) = BoundingBoxes(3)*BoundingBoxes(4)
end
```

Feature vectors

```
Areas_vector = [imagecomponents.Area]

MajorAxisLengths_vector = [imagecomponents.MajorAxisLength]

MinorAxisLengths_vector = [imagecomponents.MinorAxisLength]

BoundingBoxes_Areas_vector = [BoundingBoxes_Areas]
```

Objects matrix

```
Components_matrix = [Areas_vector MajorAxisLengths_vector ...
    MinorAxisLengths_vector  BoundingBoxes_Areas_vector]
```

Objects vectors

```
object_matrixx=cell(1,Number_objects);



for i = 1:Number_objects

    Objects_Vector_ = [Areas_vector(i) MajorAxisLengths_vector(i)...
        MinorAxisLengths_vector(i) BoundingBoxes_Areas_vector(i)];
```

```
        eval(['Objects_Vector_' num2str(i) '= Objects_Vector_'])
end
```

Published with MATLAB® R2015a

APPENDIX 2: LIST OF PUBLICATIONS

The following are the publications by the author in the course of the research period:

1. D. A. Osuto, H. A. Ouma and E. N. Ndungu, ”Vision Based Road Traffic Density Estimation and Vehicle Classification for Stationary and Moving Traffic Scenes during Daytime,” Journal of Sustainable Research in Engineering, vol. 2, no. 3, pp. 100-110, 2015.

2. D. A. Osuto, H. A. Ouma and E. N. Ndungu, ”Automatic Road Traffic Density Estimation Using Image Processing Algorithms,” in Sustainable Research and Innovation (Sri) Conference, Nairobi, Kenya, 2016.