IMPLEMENTING AN IMPROVED INTELLIGENT LICENCE PLATE DETECTION SYSTEM USING IMAGE PROCESSING AND PATTERN RECOGNITION ALGORITHMS

By

JAWAD MUHAMMAD

Submitted to the Institute of Graduate Studies in Science and Engineering

in partial fulfilment of

the requirements for the degree of

Master of Science

in

Computer Engineering

Mevlana (Rumi) University

2014

IMPLEMENTING AN IMPROVED INTELLIGENT LICENCE PLATE DETECTION SYSTEM USING IMAGE PROCESSING AND PATTERN RECOGNITION ALGORITHMS

Submitted by **Jawad Muhammad** in partial fulfilment of the requirements for the degree of Master of Science in Computer Engineering at Mevlana (Rumi) University

APPROVED BY:

Examining Committee Members:

Prof. Dr. Bekir KARLIK . . . . . .

Assoc. Prof. Dr. Halis ALTUN . . . . . .
(Thesis Supervisor)

Assist. Prof. Dr. Alaa ELEYAN . . . . . .

Associate Prof. Dr. Halis ALTUN . . . . . .
Head of Department, Computer Engineering

Assoc. Prof. Dr. Ali SEBETCI . . . . . .
Director, Institute of Graduate Studies in Science and Engineering

DATE OF APPROVAL (___/___/___)

DECLARATION

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Jawad, Muhammad:

Signature:

ABSTRACT

IMPLEMENTING AN IMPROVED INTELLIGENT LICENCE PLATE RECOGNITION SYSTEM USING IMAGE PROCESSING AND PATTERN RECOGNITION ALGORITHMS

Jawad Muhammad

Computer Engineering M.S. Thesis, 2014

Thesis Supervisor: Assoc. Prof. Dr. Halis ALTUN

Keywords: Plate Detection, HOG, License Plate, Neural Network, Genetic Algorithm

In recent years, there had been a substantial increase in demand of an efficient implementation of Automated License plate recognition (ALPR) system in countries all over the world with the system been commercially developed, deployed and used in cities as a tool for mass security surveillance. An implementation of ALPR system involves three processes: location of the license plate region; segmentation with extraction of the plate characters; and recognition of the extracted plate characters. This thesis research focuses on the first process in implementing ALPR, which is location of the license plate characters.

In this research, an algorithm is proposed that adopt the concept of Histogram of Oriented Gradients (HOG) into developing a custom algorithm for the feature extraction of a license plate portion from an image by taking into consideration the structure of the plate. Artificial Neural Network (ANN) was employed to classify and assign similarity scores to any portion selected and Genetic Algorithm (GA) samples, selects and evaluates portions using the similarity scores at which it finally select the most likely portion as the plates. Three major contributions were made by this research: the successful use of HOG in feature extraction of a plate; development of a proposed structured HOG for plates; and the successful use of the combination of GA, HOG and ANN algorithms in plate detection process.

The ANN was trained with Turkish manually cropped license plates. The plate detection was tested on a database of car images containing Turkish plates and on a database of car images with Non-Turkish plates with varying plate sizes, various environmental conditions and a lot low quality plate images. 1526 plates out of 1572 plates in the car images (97.03 %) where successfully detected for the car images with Turkish license plates and 601 plates out of 655 plates in the car images (almost 92%) where successfully detected for the car images with Non-Turkish license plates.

Dedicated to my beloved parents Alhaji Muhammad Ishaq and Hajiya Khadijah and to my late two brothers Rulai and Marwana

ACKNOWLEDGEMENTS

All praise be to Allah the beneficent and merciful. May the peace, mercy and blessings be upon his prophet Muhammad (SAW).

I wish to express my deepest gratitude to Assoc. Prof. Dr. Halis ALTUN for his guidance, advice, encouragement and insight throughout the research.

My utmost respect, love and gratefulness go to my parents Alhaji Muhammad Ishaq and Hajiya Khadija for their priceless support and understanding throughout this study. To my entire family for their patients and relentless effort on encouraging me to continue this pursuit of knowledge up to the end, specifically, my late two brothers Rulai and Marwana and my causin Lawan whom I could not share this moment with, as they get back to our lord while I was away busy with this study.

This study was fully supported by the Kano state government of Nigeria under the leadership of His Excellency Engr. Dr. Rabiu Musa Kwankwaso from September 2012 – June 2014.

TABLE OF CONTENT

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS/ABBREVIATIONS

Symbol	Explanation
GA	Genetic Algorithm
ANN	Artificial Neural Network
HOG	Histogram of Oriented Gradient
ALPR	Automatic License Plate Recognition System
PCA	Principal Components Analysis
SVM	Support Vector Machine
LDA	Local Discriminant Analysis
SIFT	Scale-Invariant Feature Transform
LBP	Local Binary Pattern
SURF	Speeded Up Robust Features

CHAPTER ONE
INTRODUCTION

1 Background

Automated License plate recognition (ALPR) system involves the detection, extraction and recognition of vehicle license plate. It can also be called: Automatic number plate recognition (ANPR); Automatic license-plate reader (ALPR); Automatic vehicle identification (AVI); Car plate recognition (CPR); License-plate recognition (LPR); Mobile license-plate and reader (MLPR). It had been an active area of research in the field of Intelligent Transportation System since 1990s when the real time ALPR systems were first developed [1-4]. ALPR systems are of paramount important due to their use in applications such as:

- Automatic location of a vehicle recognition,
- Automatic electronic payment (parking fee payment and toll payment),
- Automatic calculation of traffic volume,
- Traffic surveillance,
- Finding of stolen vehicles, access control, etc.
- Automobile repossessions [5].
- Targeted advertising [6].
- Analyses of travel behaviour (route choice, origin-destination etc.) for transport planning purposes [7].

Moreover, an increased need for security awareness has made the need for vehicle based authentication technologies extremely significant. However, these systems been generally outdoor systems that could be installed in any kind of environmental conditions encounter difficulties such as:

- variation in illumination,
- angle of view,
- vehicle moving speed,
- background complexity,
- nature and type of the vehicle,
- Wide variation in the license plate structure, background color, etc.

As such, an effective and efficient ALPR can only be achieved if all of these problems are objectively tackled during the design and development stage. Additionally, ALPR should be able to be fast in order to meet the basic demand of an intelligent transportation system.

1.1 Problem Description

In recent years, there had been a substantial increase in demand of an efficient ALPR system in countries all over the world; with ALPR been commercially developed, deployed and used in cities as a tool for mass security surveillance [8]. Commercial ALPR systems have since been developed. However, although high recognition rates was reported by the manufacturers of these systems, misidentification and high error rates were equally reported by the experience users of these systems as stated in a 2008 article [9]. Circumvention techniques by vehicle owners such as increasing the reflective properties of the lettering of the license plate, attempt to smear their license plate with dirt or utilize covers to mask the plate, etc. also present competitive challenges to the overall performance of ALPR systems as reported in [10]. Researchers had proposed algorithms which manage to tackle these problems as outlined in a survey on ALPR technologies [11]. However, there is still need for a better and more efficient algorithm which collectively addresses the issue of high misclassification, slow execution time, unfavourable nature of license surfaces, complex nature of vehicles, inconsistency in the structure of plate numbers, etc.

This work proposed an algorithm by adopting the concept of Histogram of Oriented Gradients [12], Neural Network and Genetic Algorithm into developing a custom algorithm for the detection of a license plate.

1.2 Previous Work

Research in ALPR systems had undergone tremendous break through with already a fully driven real-time ALPR applications in operation in cities all over the world. It is been an active research for the past two decades.

Research work in ALPR can be categorized into three basic processes as noted in [11]. These processes are:

- location of the license plate region in a scene image;

- segmentation and extraction of the plate characters; and
- recognition of each of the recognised character

Most of the related work on ALPR involves a combination of one or more of these processes. As already noted, in this work, the major contribution is on the location of license plate region process, as such, related works to be outlined in this section specifically involve plate localisation process.

With binary image processing, very good results were accomplished with techniques based upon combinations of edge statistics and mathematical morphology [13] [14] [15] [16]. As the change in brightness of the plate region tends to be relatively high and more regular than regions elsewhere, parameters such as gradient magnitude and variance are computed and used in edge based methods. The major disadvantage of edge based methods is that, in complex images with many unwanted edges, they can hardly be applied as they are very sensitive to unwanted edges. However, edge based methods when combined with morphological steps that eliminate unwanted edges in the processed images; the license plate localisation accuracy is relatively high and fast, compared to other methods. In [15] Zheng, D. et al. (2005) and [16] Wang, S. et al. (2003) propose plate localisation based on edge statistics. Martin, F., et al. (2002) proposes a mainly morphological operation to detect the licence plate which was termed *Top Hat* [29]. In [14] Hongliang et al. (2004) a hybrid system for detection and segmentation of a plate number that employs both edge statistics and morphological operation was proposed.

Anagnostopoulos et al. (2006) proposed a sliding concentric windows technique in segmenting the plate region of an image from the scene image using threshold value as the deciding criteria for the separation [17]. Shyang-Lih et al. (2004) proposed a plate detection by using a set of fuzzy disciplines attempts to extract license plates from an input image [18]. Azad, R. et al. (2013) propose a method of plate detection by conversion of the license plate into HSV colour space and using density and geometry of divided regions within the image to detect plates [19].

Louka Dlagnekov (2004) find the vertical and horizontal derivatives of the plate image as features and uses a classifier trained with Adaboost to classify parts of an image within a search window as either license plate or non-license plate [20]. Ho, W. T. et al. (2009) proposed a two stage license plate detection method by using Adaboost in first stage followed

by SIFT with SVM features in the second stage [21]. Assis da Silva, et al. (2013) proposed SIFT features with template matching in detecting a plate region of a particular image [22].

Although, a lot of related work had been highlighted in this section, there exists a comprehensive survey on the trends and technological advancement in the research of building ALPR systems by Anagnostopoulos, et al. (2008) [11].

This work will focus on how to implement, customise or develop pattern recognition and image processing software algorithms for performing an automated task of accurately recognising license plate numbers.

1.3 Scope and Outline

This thesis discuss in detail the theoretical background, design methodologies and implementation details of said research project (ALPR plate detection or plate localisation process and related processes from an image captured by a camera). It is aimed at providing the complete technical details needed to fully re-implement the stated research in this thesis if the need may arise. It involves step by step description of each algorithm developed in this research with explanatory schematic diagrams depicting the visual presentations of different processes.

The thesis is organised as follows:

- **Chapter One** covers the research introduction, problem definition and review of some of the research related work.
- **Chapter Two** explains the related theoretical background behind the techniques, methodologies or algorithms employed in this research.
- **Chapter Three** encompass general overview of a typical ALPR system; the units that made up the building blocks of the ALPR system are fully discussed.
- **Chapter Four** depicts the full technical design details of all the algorithms and methodologies involved or developed in this research.
- **Chapter Five** discuss the details of experiments carried out by this research, results of the experiments, and evaluation with analysis of these results.
- **Chapter Six** is where conclusion about the whole research is stated and recommendations for future research are expanded.

CHAPTER TWO
THEOROTICAL BACKGOUND

2 Introduction

ALPR systems are built based on concepts from so many engineering research study fields such as Artificial Intelligence, Machine Learning, Pattern Recognition, Image Processing and Mathematical Statistical concepts. In this chapter, a detailed theoretical background of some of these concepts are explained, with emphasis on the theories directly related, employed or adopted by this project.

As introduced in the previous chapter, this work adopts concepts from image processing, artificial intelligence and pattern recognition. Specifically, methods of image processing techniques such as image gradients, histogram vector, image orientation, image feature extraction technique, etc. were directly employed by this project. Also, Artificial Intelligence methods of data classification and searching optimization such as Neural Network, Support Vector Machine, and Genetic Algorithm were also used.

The theories explained in this chapter are divided into two (2) major categories;

- Theories related to image processing and
- Theories related to pattern recognition.

Details of each of these theories with some other supplementary information will be uncovered in the subsequent sections of this chapter.

2.1 Image Processing

An image may be define as a two-dimensional function, $f(x, y)$, where x and y are spatial coordinates [23]. Intensity or grey level of an image refers to the amplitude of f at any pair of coordinates (x, y) [23]. Digital image is the image of which the values of x, y and intensity are all finite, discrete quantities and a finite number of elements. The field of image processing or digital image processing refers to processing digital images by means of a

digital computer [23]. A Digital image is normally represented as a matrix of n rows and m columns with the notation $n \times m$ used to indicate the size of the image as shown in Fig. 2.1 All matrix operations can be performed on a digital image, this result to numerous opportunities in the manipulation of a digital image.

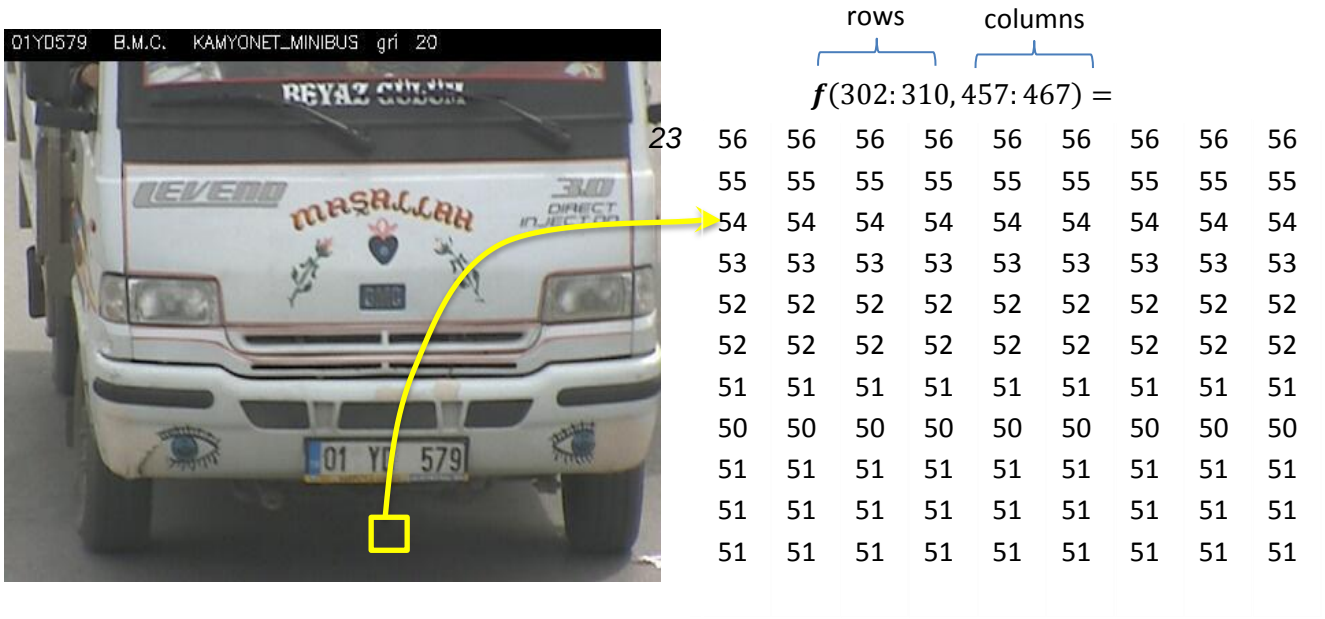


Figure 2.1. Digital Image Representation

Image processing process typically comprises of three basic components or stages as shown in Fig. 2.2:

- Image Capture (Input)
- Image Processing
- Output

Figure 2.2. Image Processing Basic Components or sequential stages

The first step of image processing is the image capturing process; this is simply the use of an image sensing device such as a camera device or its equivalent to capture the picture of a world scene and convert the captured picture into a digital image. Ones an image is in a digital form, appropriate image processing techniques can be applied to this image which will either transform or change the image composition or even extract some useful information from this image. The output of image processing could be an image, statistical results or combination of some parts or features of the originally processed image.

Numerous image processing techniques had been devised in the literature starting from a simple image addition, subtraction or even multiplication to more complex techniques such as image transformations, image filtering, image gradients, morphological operations, etc. in this work, notable image processing techniques used are:

- Spatial filtering and image enhancement
- Gradient vector
- Image feature extraction

2.1.1 Spatial Filtering and Image Enhancement

Image captured from a natural scene especially in outdoor environment tends to be far from been perfect in its structural composition as factors such as occlusion, hash weather condition, relative movement of camera with objects, etc. lead to degradation of the image with noise, hence, this necessitate the use of techniques such as spatial filtering in order to improve the quality of the image. Spatial filtering involves performing operation on a pixel of an image based on the condition of its neighbouring pixels. Equation (2.1) and Fig. 2.3 indicates a typical operation of a spatial filter.

$$g(x, y) = T[f(x, y)] \quad (2.1)$$

Where $f(x, y)$ is the intensity of the pixel at point (x, y) and $g(x, y)$ is the new intensity value of the pixel at that point after the application of the spatial filter,T

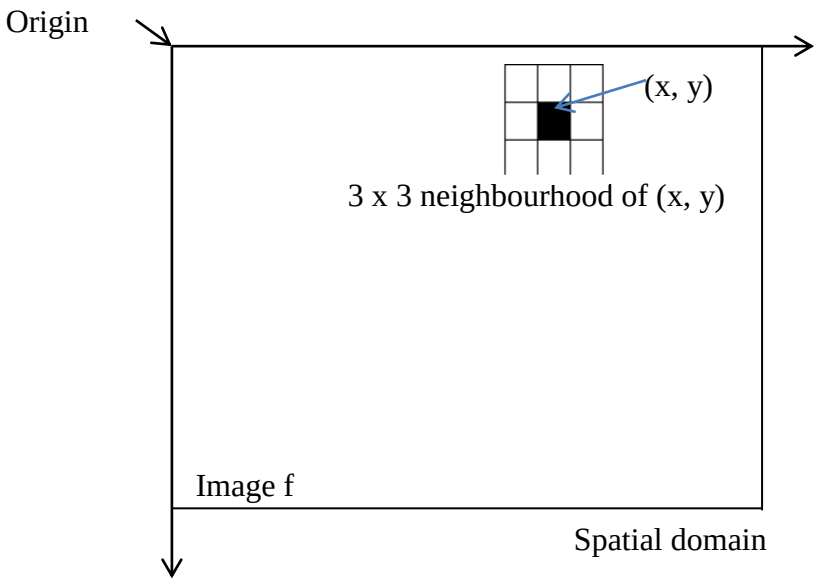


Figure 2.3. Spatial Filtering process, a 3 × 3 neighbourhood about a point $(\mathbf{x}, \mathbf{y})$ in an image in the spatial domain [23]

The process described in Fig. 2.3 consists of moving the origin of the neighbourhood from pixel to pixel and applying the operator T to the pixels in the neighbourhood to yield the output at that location. Thus, for any specific location (x, y), the value of the output image g at those coordinates is equal to the result of applying T to the neighbourhood with origin at (x, y) in f. The operator T could be an averaging operator where by the entire neighbourhood pixels are averaged together with the pixel in question or addition operator, subtraction operator, etc.

Although spatial filtering can be applied to broad range of applications, it is predominantly useful in image enhancement. Image enhancement is the process of manipulating an image so that the result is more suitable than the original image for a specific application [23]. In most image processing applications, image enhancement is done at the pre-processing stage where by the quality of the original image is first improved through image enhancement before proceeding to the later stage of the image processing. In this work, during the pre-processing stage, image enhancement was performed on the original image which involves application of an averaging filter to the vertical gradient of the original image in order to spread the effect of the edges of objects such as text around the neighbouring pixel for the purpose of achieving image localisation.

2.1.2 Image Gradients

Image gradients or simply gradient vector is the measure of the rate of change in the pixel values of an image along x-direction and along y-direction. Computing image gradients is mainly a filtering operation such as correlation or convolution operation where by a filter kernel corresponding to the gradient operator is scanned over an image in one pass and the gradient values of each pixel along the x and y directions are obtained using Equation (2.2), (2.3) and (2.4) as showed in Fig. 2.4.

$$\nabla f = grad(f) = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{df}{dx} \\ \frac{df}{dy} \end{bmatrix} \quad (2.2)$$

$$M(x,y) = mag(\nabla f) \approx |g_x| + |g_y| \quad (2.3)$$

$$\text{Orientation}(x,y) = \text{angle}(\nabla f) = \tan^{-1}\left[\frac{g_x}{g_y}\right] \quad (2.4)$$

Where g_x and g_y is the gradient vector along x and y direction respectively, computed using one of the gradient operators.

In Fig. 2.4 a simple gradient operator was used, in theory; there are more complex gradient operators such as Roberts cross gradient operator, Sobel operator, Laplacian operator, etc. and two of them shown in Fig. 2.5 and Fig. 2.6. Computation of the gradient using these operators is simply by sliding the operators as a filter over the image and calculating corresponding correlation values at each pixel.

Gradient vectors are mostly used for the detection of edges and feature extraction of an image. In this work, Sobel gradient operator was used for the detection of pixel edges and in the feature extraction process.

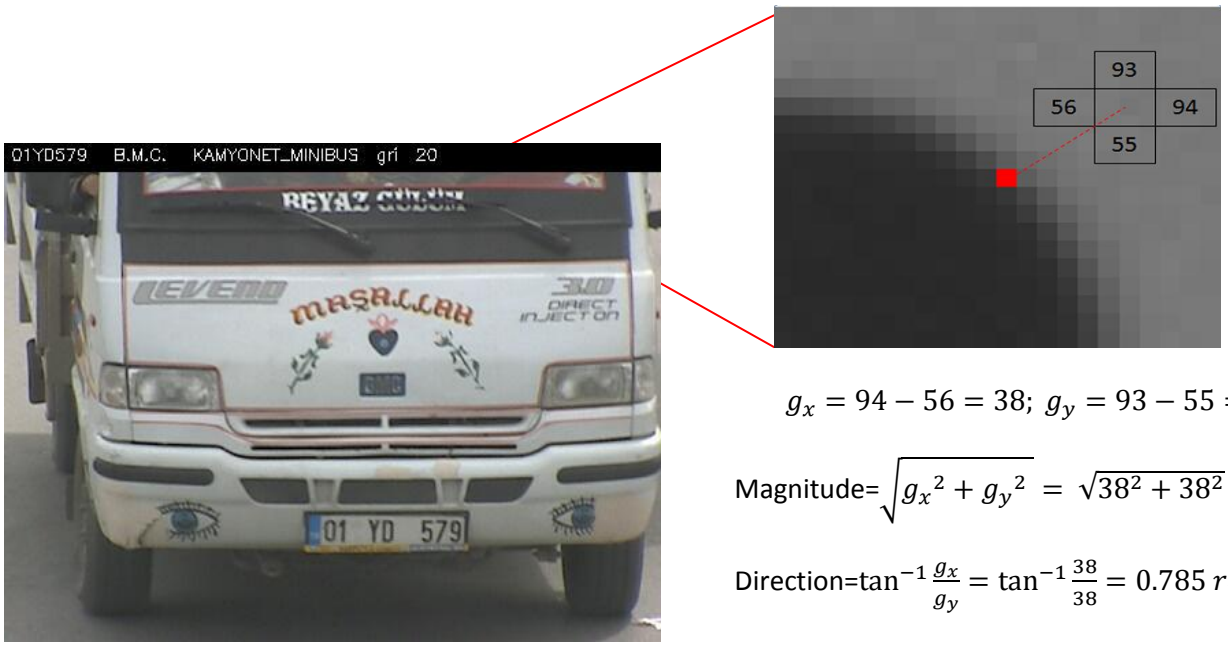


Figure 2.4: Gradient Computation process; the red rectangle represent a selected pixel with 4 immediate neighbours (93, 94, 55 and 56); $\mathbf{g_x}$ and $\mathbf{g_y}$ are the gradient of the choosing pixel along **x** and **y** direction respectively.

0	-1
1	0

-1	0
0	1

Figure 2.5: Roberts cross operators

-1	0	1
-2	0	2
-1	0	1

-1	-2	-1
0	0	0
1	2	1

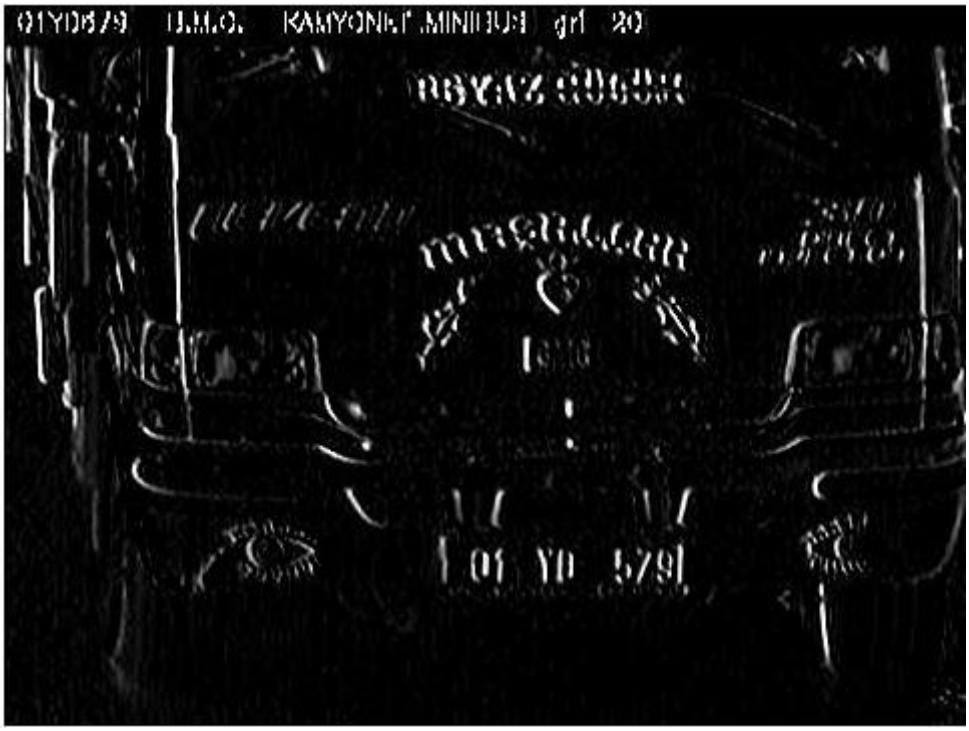
Figure 2.6: Sobel operators

In Fig. 2.7 Sobel edge operator is used in the detection of edges. In (a), the grey image is shown, (b) shows the gradient along x-direction, (c) shows the gradient along y-direction and (d) shows the combined magnitude gradient image.


01YD579 B.M.C. KAMYONET_MINIBUS gri 20
BEYAZ
MASALLAH
30
DIRECT
INJECTION
01 YD 579

(a)

(b)



(c)

(d)

Figure 2.7. (a). Images of the Grey image and the gradient image computed using the Sobel operator (b) Along the X-direction (c) Along the Y-direction (d) Gradient magnitude image.

2.1.3 Image Feature Extraction

Image feature extraction involves the use of image processing techniques to detect and isolate various desired portions or shapes (features) of a digitized image or video stream [23]. Image feature extraction can be categorised into *low level feature extraction* and *high level feature extraction* [24].

Low-level features can be defined as those basic features that can be extracted automatically from an image without any shape information (information about *spatial* relationships) [24]. They are normally the results of performing point operation on an image where by each pixel of the image is considered independently during the processing of the image. As such, processes such as image thresholding, image gradient and edge detection, are forms of low-level feature extraction.

High-level features are the features that concerns finding shapes in computer images. To be able to recognize faces automatically, for example, one approach is to extract the component features. This requires extraction of, say, the eyes, the ears and the nose, which are the major facial features. To find them, we can use their shape: the white part of the eyes is ellipsoidal; the mouth can appear as two lines, as do the eyebrows. Shape extraction implies finding their position, their orientation and their size.

In feature extraction, we generally seek *invariance properties*, most notable *invariance properties* in an image are:

- ✓ Illumination
- ✓ Position, location or translation invariance
- ✓ Rotation or orientation invariance

As a basic *invariant*, we seek immunity to changes in the *illumination* level: we seek to find a shape whether it is light or dark. In principle, as long as there is contrast between a shape and its background, the shape can be said to exist, and can then be detected. (Clearly, any computer vision technique will fail in extreme lighting conditions; you cannot see anything when it is completely dark.) Following illumination, the next most important parameter is *position*: we seek to find a shape wherever it appears. This is usually called *position*, *location* or *translation invariance*. Then, we often seek to find a shape irrespective of its *rotation* (assuming that the object or the camera has an unknown orientation); this is usually called *rotation* or *orientation invariance*. Then, we might seek to determine the object at whatever

size it appears, which might be due to physical change, or to how close the object has been placed to the camera. This requires *size* or *scale invariance*. These are the main invariance properties we shall seek from our shape extraction techniques. However, nature (as usual) tends to roll balls under our feet: there is always *noise* in images. In addition, since we are concerned with shapes, there may be more than one in the image. If one is on top of the other it will *occlude*, or hide, the other, so not all of the shape of one object will be visible.

There exists numerous feature extraction techniques, among them are:

- ✓ Thresholding and subtraction;
- ✓ Principle components analysis (PCA) [25];
- ✓ Local discriminant analysis (LDA) [26];
- ✓ Local Binary patterns (LBP) [27];
- ✓ Histogram of Oriented Gradient (HOG) [12]
- ✓ Speeded Up Robust Features(SURF) [28]
- ✓ Scale-invariant feature transform (SIFT) [29]
- ✓ Etc.

Among all these techniques, Histogram of Oriented Gradient (HOG) feature extraction technique is the only directly related technique used in this research, as such, the next section will be devoted to the in-depth study of this technique.

2.1.3.1 Histogram of Oriented Gradient (HOG)

HOG are image feature descriptors employed in computer vision and image processing for the purpose of object detection or pattern recognition as proposed by Dalal, N., & Triggs, B. (2005) [12]. The technique counts the occurrences of gradient orientation in localised portions of an image. The basic idea behind the HOG descriptors is that local object appearance and shape within an image can be defined by the distribution of intensity gradients or edge directions. The implementation of these descriptors can be achieved by dividing the image into small connected regions, called cells, and for each cell compiling a histogram of gradient directions or edge orientations for the pixels within the cell. The combination of these histograms then represents the descriptor. For improved performance, the local histograms can be contrast-normalized by calculating a measure of the intensity across a larger region of

the image, called a block, and then using this value to normalize all cells within the block. This normalization results in better invariance to changes in illumination or shadowing.

HOG implementation involves gradient computation; orientation binning and descriptor blocks Normalisation. The steps are as follows:

- ✓ The first step in implementing HOG feature extraction is the Gradient computation which was discussed in detail in the previous section. At this stage, the gradient magnitude image and gradient orientation is obtained.
- ✓ The next step is the division or grouping of the image pixels into cells regions and calculating or generating cell histograms vector for each of these regions. Each pixel within the cell casts a **weighted vote** for an orientation-based histogram channel based on the values found in the gradient computation. The cells themselves are rectangular and the histogram channels are evenly spread over 0 to 180 degrees or 0 to 360 degrees, depending on whether the gradient is unsigned or signed. Dalal and Triggs [12] found that unsigned gradients used in conjunction with 9 histogram channels performed best in their experiments. As for the vote weight, pixel contribution can be the gradient magnitude itself, or the square root or square of the gradient magnitude. Fig. 2.8 and Fig. 2.9 shows the previously obtained magnitude image divided into cells and cell histogram vector is computed in each of these cells in form of bins.
- ✓ The final step in HOG computation is the local descriptor block normalisation of which the cell histogram vectors computed above are concatenated into a single HOG descriptor vector. In order to account for changes in illumination and contrast, the gradient strengths must be locally normalized, which requires grouping the cells together into larger, spatially-connected blocks. The HOG descriptor is then the vector of the components of the normalized cell histograms from all of the block regions. These blocks typically overlap, meaning that each cell contributes more than once to the final descriptor. Two main block geometries exist: rectangular R-HOG blocks and circular C-HOG blocks. R-HOG blocks are generally square grids, represented by three parameters: the number of cells per block, the number of pixels per cell, and the number of channels per cell histogram. Any of the normalisation method could be used but the most commonly used and effective methods is the *L1* normalisation and *L2* normalisation.

Figure 2.8. Car gradient magnitude image divided into cells regions

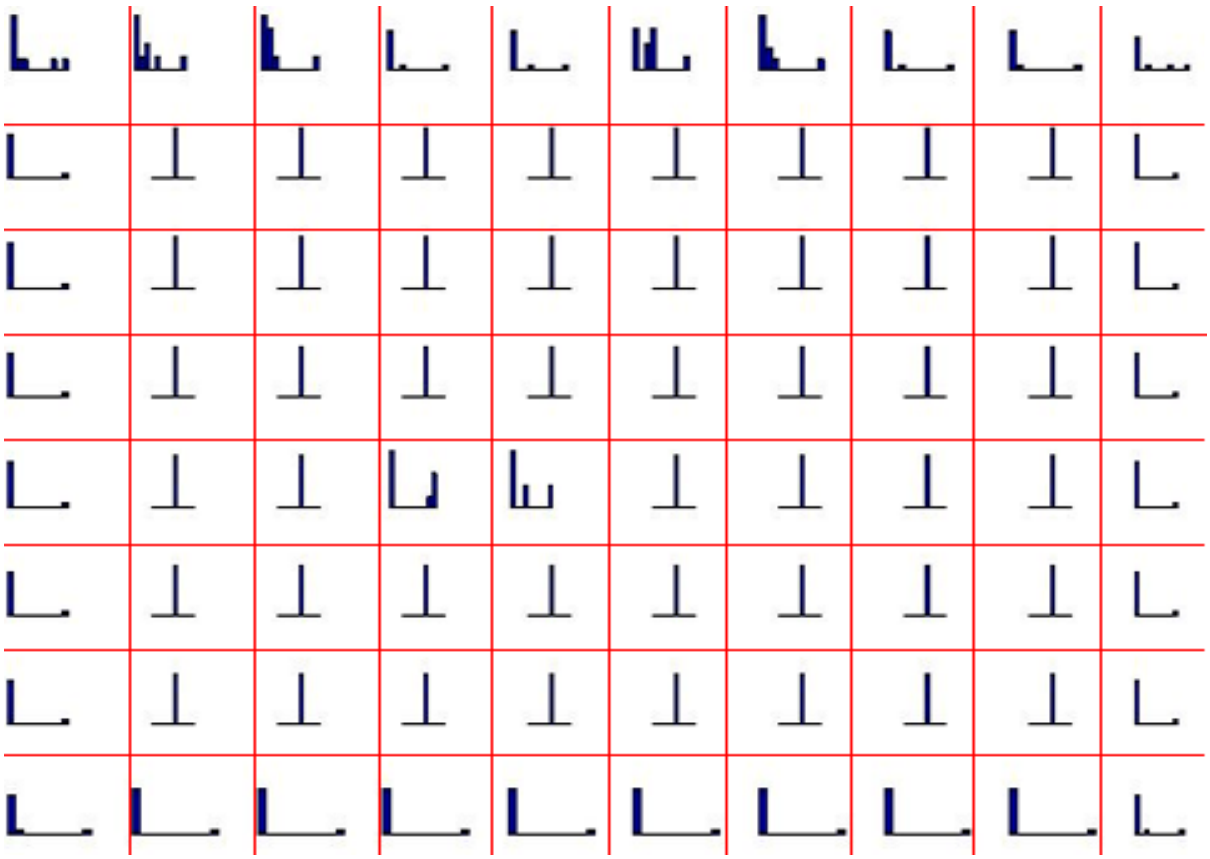


Figure 2.9. Histogram computed over each cell division

2.2 Pattern Recognition

Pattern recognition involves the assignment of a label to a given input value. Pattern recognition algorithms generally aim to provide a reasonable answer for all possible inputs and to perform *most likely* matching of the inputs, taking into account their statistical variation. One major category of pattern recognition algorithms is *classification*, which attempts to assign each input value to one of a given set of classes (for example, determine whether a given image is "plate number" or "non-plate number"). Pattern classification algorithms can be divided into two categories:

- Supervised and
- Unsupervised classification.

A classification procedure is said to be *unsupervised* if no training data are required and the user needs to specify the number of classes (at most) [30]. Example of unsupervised algorithms is the data clustering algorithms such as EM algorithm.

A classification procedure is said to be *supervised* if the user either defines the decision rules for each class directly or provides training data for each class to guide the computer classification [30]. All the pattern recognition algorithms employed in this research are supervised pattern recognition classification algorithms. Some of the most commonly used pattern recognition classification algorithms are:

- Support vector machines (SVM) [31].
- Artificial neural network (ANN) [32].
- Bayesian Network [33].
- Fuzzy logic based classification [34].
- Adaptive boosting [35].
- Etc.

ANN is the pattern recognition classifiers that is directly related to this research; hence, it details will be unveiled in the sub-sequent sections.

Another important technique in pattern recognition for performing systematic searching of a specific pattern in complex patterns is the *Genetic Algorithm (GA)*. It is mostly used in solutions optimization problems. GA will also be discussed in detail in the upcoming section as it is also employed in this research.

2.2.1 Artificial Neural Network

Artificial neural network (ANN) provides a robust approach to approximating real-valued, discrete-valued, and vector-valued target functions [30]. Artificial Neural Networks are composed of network of nodes or units (see Fig. 2.10) connected by directed links**.** A *link* from unit j to unit i serve to propagate the activation a_j from j to i. Each link also has a numeric weight $w_{j,i}$ *a*ssociated with it, which determine the strength and sign of the connection. Each unit i first computes a weighted sum of its inputs (see Equation (2.5)), and then it applies an activation function g to this sum to derive the output (see Equation (2.6)) [36].

$$in_i = \sum_{j=0}^{n} w_{j,i} a_j \quad (2.5)$$

$$a_i = g(in_i) = g\left(\sum_{j=0}^{n} w_{j,i} a_j\right) \quad (2.6)$$

Where $a_1, a_2, a_3, a_4, \ldots\ldots\ldots a_n$ are referred to as *real* inputs, $w_1, w_2, w_3, w_4, \ldots\ldots\ldots w_n$ are their corresponding *real* weights, a_0= -1 is the bias input and w_0 is the bias weight, g() is an activation function.

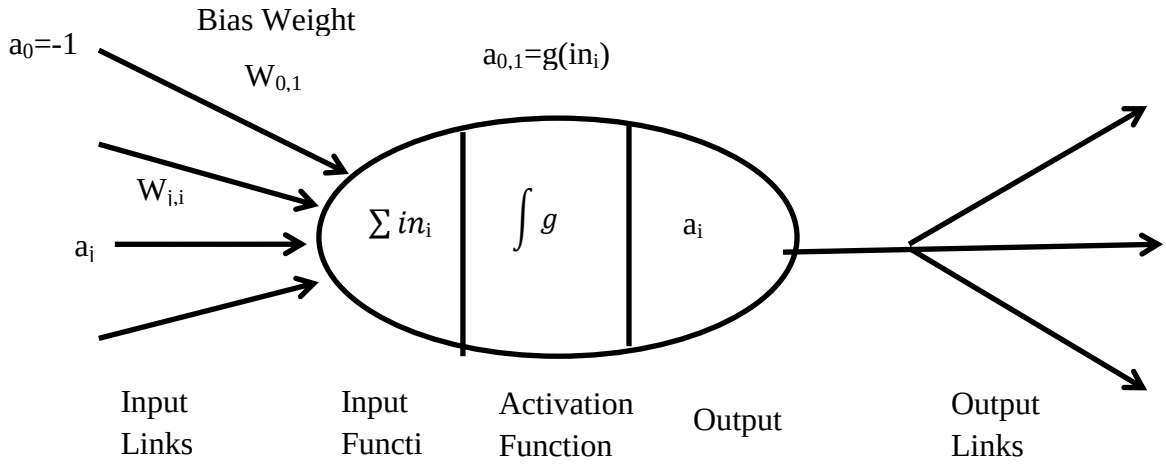


Figure 2.10: A simple mathematical model for a unit. The unit's output activation is $\mathbf{a_i} = \mathbf{g}(\sum_{\mathbf{j=0}}^{\mathbf{n}} \mathbf{w_{j,i}} \mathbf{a_j})$, where a, is the output activation of unit j and $\mathbf{w_{j,i}}$ is the weight on the link from unit j to this unit.

The activation function should be able to meet the following criteria:

- Be *active* (near +1) when the right inputs are given and *inactive* (near 0) when wrong inputs are given.
- Needs to be non-linear

In Fig. 2.11, two choices of activation function are shown; *threshold* function and *sigmoid* function. The *threshold* function has a *hard* threshold at zero while the *sigmoid* has a *soft* threshold at zero; both of the functions are activated when the weighted sum of *real* inputs exceeds the bias input (see Equation (2.7)).

$$a_i => +1 => \text{active} => \sum_{j=1}^{n} w_{j,i}a_j > w_{0,i} \qquad (2.7)$$

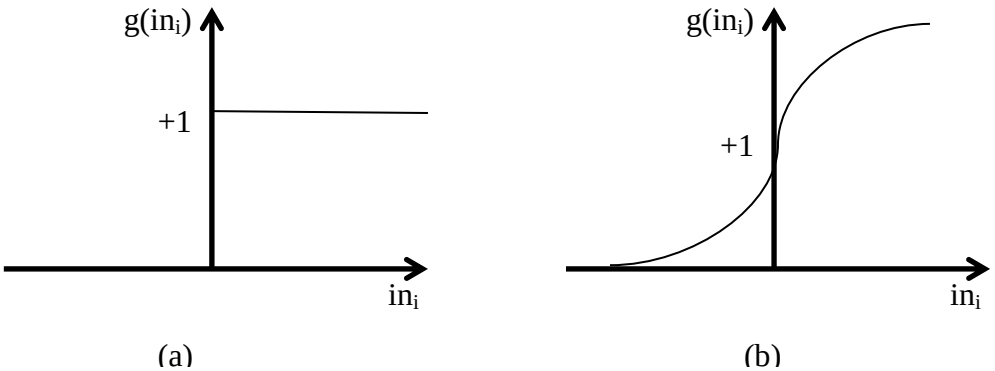


Figure 2.11. (a) the threshold activation function, which outputs 1 when the input is positive and 0 otherwise. (Sometimes the sign function is used instead, which outputs $\pm\mathbf{1}$ depending on the sign of the input.) (b) The sigmoid function.

There are two main categories of ANN structure:

- Acyclic or feed-forward networks and
- Cyclic or recurrent networks

A *feed-forward network* represents a function of its current input; thus, it has no internal state other than the weights themselves. A *recurrent network*, on the other hand, feeds its outputs back into its own inputs. This means that the response of the network to a given input depends on its initial state, which may depend on previous inputs. Hence, recurrent networks (unlike feed-forward networks) can support short-term memory. This section will concentrate on feed-forward networks because it is the category most relevant to this research.

2.2.1.1 Feed-Forward Network

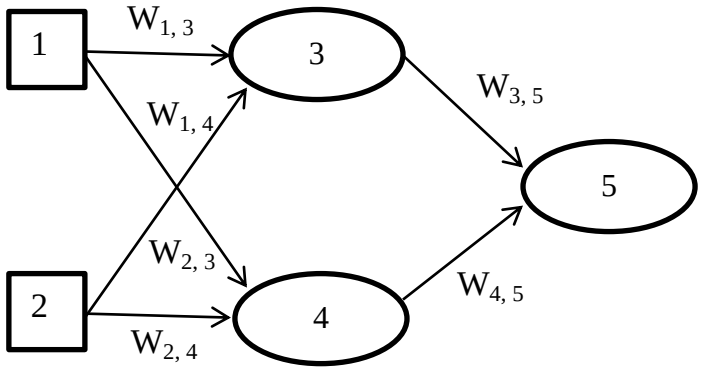


Figure 2.12. A very simple neural network with two inputs, one hidden layer of two units, and one output

Consider the simple network shown in Fig 2.12, which has two input units (1 and 2), two *hidden* units (3 and 4)**,** and an output unit (5). Any unit that is not among the input or the output unit is normally referred to as *hidden* unit. Equation (2.8) represent the network equation, indicating the output a_5. This typically depicts a Feed-forward network as it is mainly represented by the value of its input.

$$a_5 = g(w_{3,5}.a_3 + w_{4,5}.a_4)$$

$$= g(w_{3,5}.g(w_{1,3}.a_1 + w_{2,3}.a_2) + w_{4,5}.g(w_{1,4}.a_1 + w_{2,4}.a_2)) \quad (2.8)$$

It can be observed that the weights in the network act as *parameters* of this function; writing w for the parameters, the network computes a function $h_w(x)$. By adjusting the weights, the function that the network represents will be changed. This is how learning occurs in ANN.

Learning in ANN start by passing to the network series of training examples (set of inputs values) and the network iteratively readjust its weights until a stopping criteria is reached, at which the network is said to have been trained. A trained ANN is what will be used in the classification of an unknown input.

Feed-forward ANNs are usually arranged in layers**,** such that each unit receives input only from units in the immediately preceding layer. Based on this, two categories of Feed-forward ANNs can be devised:

- Single layer (perceptron) and
- Multilayer

In *single-layer* or *perceptron* Feed-forward ANN, all the inputs are connected directly to the outputs. While *multilayer* Feed-forward ANN is when there is at least one intermediary layer between the input and the output. Since each output unit in *single-layer* is independent of the others, each weight affects only one of the outputs.

In this research, *multilayer* Feed-forward ANN was used; Fig. 2.13 depict a multilayer feed-forward ANN with one hidden layer and 10 inputs. Learning algorithms for multilayer ANN are similar to the perceptron learning algorithm, the major difference is that, whereas the $h_w(x)$ at the output layer is clear, the error at the hidden layers seems mysterious because the training data does not say what value the hidden nodes should have. It turns out that the error from the output layer can be *back-propagated* to the hidden layers, this form bases for a well-known learning algorithm of a *multilayer* feed-forward ANN called *Back-propagation*. The detail pseudo-code for *Back-propagation* algorithm is given in Table (2.1) [36].

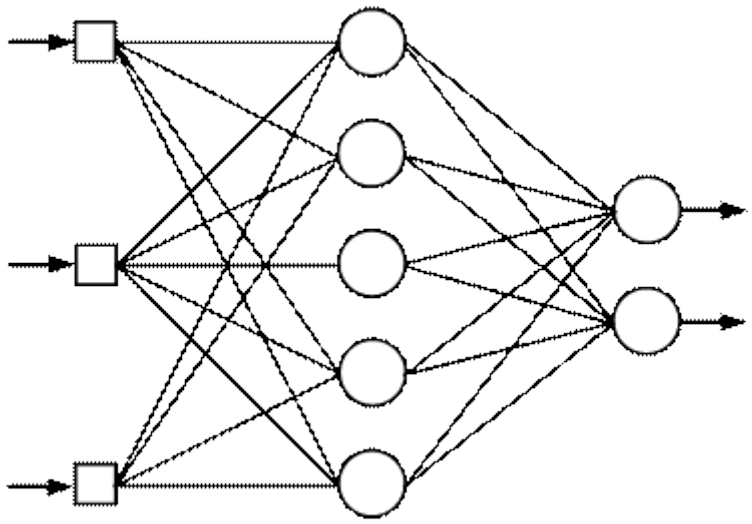


Figure 2.13. A feed forward Multi-Layer Perceptron (MLP) using sigmoid-type neurons (with 3 inputs, 5 and 2 neurons in the hidden and output layers respectively) [36]

The back-propagation process can be summarized as follows:

- Compute the ∆ values for the output units, using the observed error.
- Starting with output layer, repeat the following for each layer in the network, until the earliest hidden layer is reached:
    - Propagate the ∆ values back to the previous layer.
    - Update the weights between the two layers

Table 2.1. The back-propagation algorithm for learning in multilayer networks [36]

function B A C K - P R O P - L E A R N I N G *(examples, network)* ***returns*** *a neural network*
inputs: *examples, a set of examples, each with input vector X and output vector y;*
network, a multilayer network with ***L*** *layers, weights* $W_{j,i}$*, activation function* g*;*
repeat
for each *e* ***in*** *examples* ***do***
for each *node* j *in the input layer* ***do*** $a_j \leftarrow x_j[e]$
for l=2 ***to L do***
$in_i \leftarrow \sum_{j,i} W_{j,i}\, a_j$
$a_i \leftarrow g(in_i)$
for each *node* i *in the output layer* ***do***
$\Delta_i \leftarrow g(in_i) \times (y_i[e] - a_i)$
for l=***L- l to*** 1 ***do***
for each *node* j *in layer* l ***do***
$\Delta_j \leftarrow g(in_j) \times \sum_i W_{j,i}\ \Delta_i$
for each *node* i *in layer* $l + 1$ ***do***
$W_{j,i} \leftarrow W_{j,i} + \alpha \times a_j \times \Delta_i$
until *some stopping criterion is satisfied*
return NEURAL-NET-HYPOTHESIS*(network)*

2.2.2 Genetic Algorithm

Genetic algorithms (GAs) are forms of computational models motivated by evolution [30]. These algorithms encode a potential solution to a specific problem on a simple chromosome-like data structure, and apply recombination operators to these structures in such a way as to preserve critical information. An implementation of a GA begins with a population of (typically random) chromosomes. One then evaluates these structures and allocates reproductive opportunities in such a way that those chromosomes which represent a better solution to the target problem are given more chances to *reproduce* than those chromosomes

which are poorer solutions. The *fitness* of a solution is typically defined with respect to the current population.

A prototypical genetic algorithm [30] is described in Table (2.2). The inputs to this algorithm include the fitness function for ranking candidate solutions, a threshold defining an acceptable level of fitness for terminating the algorithm, the size of the population to be maintained, and parameters that determine how successor populations are to be generated: the fraction of the population to be replaced at each generation and the mutation rate.

Notice in this algorithm, each iteration through the main loop produces a new generation of chromosomes based on the current population. First, a certain number of chromosomes from the current population are selected for inclusion in the next generation. These are selected *probabilistically***,** where the probability of selecting chromosome *ci* is given by Equation (2.9).

$$\Pr(\mathrm{ci}) = \frac{\mathrm{Fitness(ci)}}{\sum_{\mathrm{j=1}}^{\mathrm{p}} \mathrm{Fitness(cj)}} \qquad (2.9)$$

Table 2.2. A prototypical genetic algorithm. [30].

GA*(Fitness, Fitness_threshold, p, r, m)*

Fitness: A function that assigns an evaluation score, given a hypothesis.
Fitness_threshold: A threshold specifying the termination criterion.
p: The number of chromosome *to be included in the population.*
r: The fraction of the population to be replaced by Crossover at each step.
m: The mutation rate.

- ***Initialize population:*** *P*← *Generate p* chromosomes *at random*
- ***Evaluate:*** *For each h in P, compute* $Fitness(ci)'$
- ***While*** [$\max_c Fitness(c)$] < *Fitness_threshold* ***do***

Create a new generation, Ps:

1. ***Select:*** *Probabilistically select* $(1-r)p$ *members of P to add to* P_s*. The probability* $\Pr(hi)$ *of selecting* chromosome *ci from P is given by*

$$\Pr(ci) = \frac{Fitness(ci)}{\sum_{j=1}^{p} Fitness(cj)}$$

2. Crossover: *Probabilistically select* $\frac{r.p}{2}$ *pairs of* chromosomes *from P, according to* $P(ci)$ *given above. For each pair,* $(c_1 - c_2)$*, produce two offspring by applying the Crossover operator. Add all offspring to* P_s**.**

3. ***Mutate:*** *Choose m percent of the members of* P_s *with uniform probability. For each, invert one randomly selected bit in its representation.*

4. ***Update:*** $P \leftarrow P_s$.
5. ***Evaluate:*** *for each h in P, compute* $Fitness(ci)$

- *Return the* chromosome *from P that has the highest fitness.*

The generation of successors in a GA is determined by a set of operators that recombine and mutate selected members of the current population. Typical GA operators for manipulating bit string chromosomes are illustrated in Table (2.2). These operators correspond to idealized versions of the genetic operations found in biological evolution. The two most common operators are:

- *crossover* and
- *mutation*

2.2.2.1 Crossover

The *crossover operator* produces two new offspring from two parent strings, by copying selected bits from each parent. The bit at position i in each offspring is copied from the bit at position i in one of the two parents. The choice of which parent contributes the bit for position i is determined by an additional string called the *crossover mask.* There are 3 variations of cross over operations [30].

- ❖ Single point Crossover
- ❖ Two Point Crossover
- ❖ Uniform Crossover

2.2.2.1.1 Single point crossover

To illustrate, consider the *single-point crossover* operator at the top of Table 2.3. Consider the topmost of the two offspring in this case. This offspring takes its first five bits from the first parent and its remaining six bits from the second parent, because the crossover mask 11 11 1000000 specifies these choices for each of the bit positions. The second offspring uses the same crossover mask, but switches the roles of the two parents. Therefore, it contains the bits that were not used by the first offspring. In single-point crossover, the crossover mask is always constructed so that it begins with a string containing n contiguous Is, followed by the

necessary number of 0s to complete the string. This results in n offsprings in which the first n bits are contributed by one parent and the remaining n bits by the second parent. Each time the single-point crossover operator is applied the crossover point n is chosen at random, and the crossover mask is then created and applied.

2.2.2.1.2 Two point crossover

In *two-point crossover*, offspring are created by substituting intermediate segments of one parent into the middle of the second parent string. Put another way, the crossover mask is a string beginning with $n0$ zeros, followed by a contiguous string of $n1$ ones, followed by the necessary number of zeros to complete the string. Each time the two-point crossover operator is applied, a mask is generated by randomly choosing the integers $n0$ and $n1$. For instance, in the example shown in Table 2.3 the offspring are created using a mask for which $n0 = 2$ and $n1 = 5$. Again, the two offspring are created by switching the roles played by the two parents.

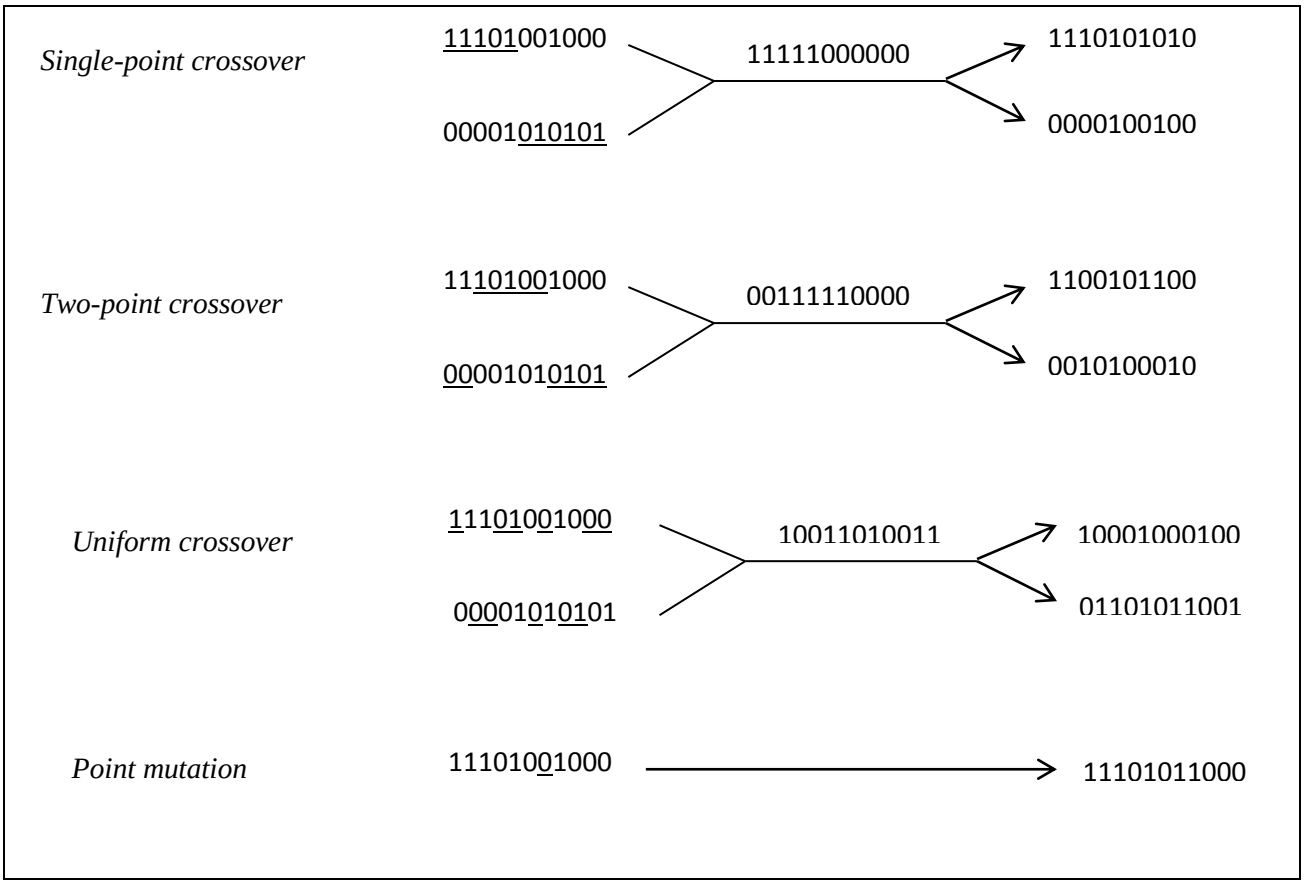
2.2.2.1.3 Uniform crossover

Uniform crossover combines bits sampled uniformly from the two parents, as illustrated in Table 2.3. In this case the crossover mask is generated as a random bit string with each bit chosen at random and independent of the others.

2.2.2.2 Mutation

In addition to recombination operators that produce offspring by combining parts of two parents, a second type of operator produces offspring from a single parent. In particular, the *mutation* operator produces small random changes to the bit string by choosing a single bit at random, then changing its value. Mutation is often performed after crossover has been applied as in the prototypical algorithm from Table 2.2.

Table 2.3. Common operators for genetic algorithms [30]



CHAPTER THREE
ALPR SYSTEM

3 Introduction

ALPR system involves the use of an image sensing device to capture the image of a vehicle that may be either moving or standing at a fixed position. Process and detect the plate number from this image, and intelligently extract the plate number characters information as described in Fig. 3.1. ALPR system uses only a camera device connected to a computer where by images captured by the camera are acquired and pass to the computer, the computer pre-process the captured image, detects the position of the plate number, extract the characters of the plate and perform other functionalities with the extracted plate number. Like storing the plate number into a database, or comparing the plate number against some pre-stored plate numbers, in other to perform vehicle specific actions or even only recognising the vehicle. ALPR system is mostly an integrated software system that incorporates all the functionalities of camera, image acquisition, image processing, and plate number pattern recognition. As already noted, the basic components of an ALPR system are:

- Image Acquisition
- Plate Detection (Plate Segmentation)
- Characters extraction (Plate Characters Segmentation)
- Character Recognition
- Plate Number Post Processing

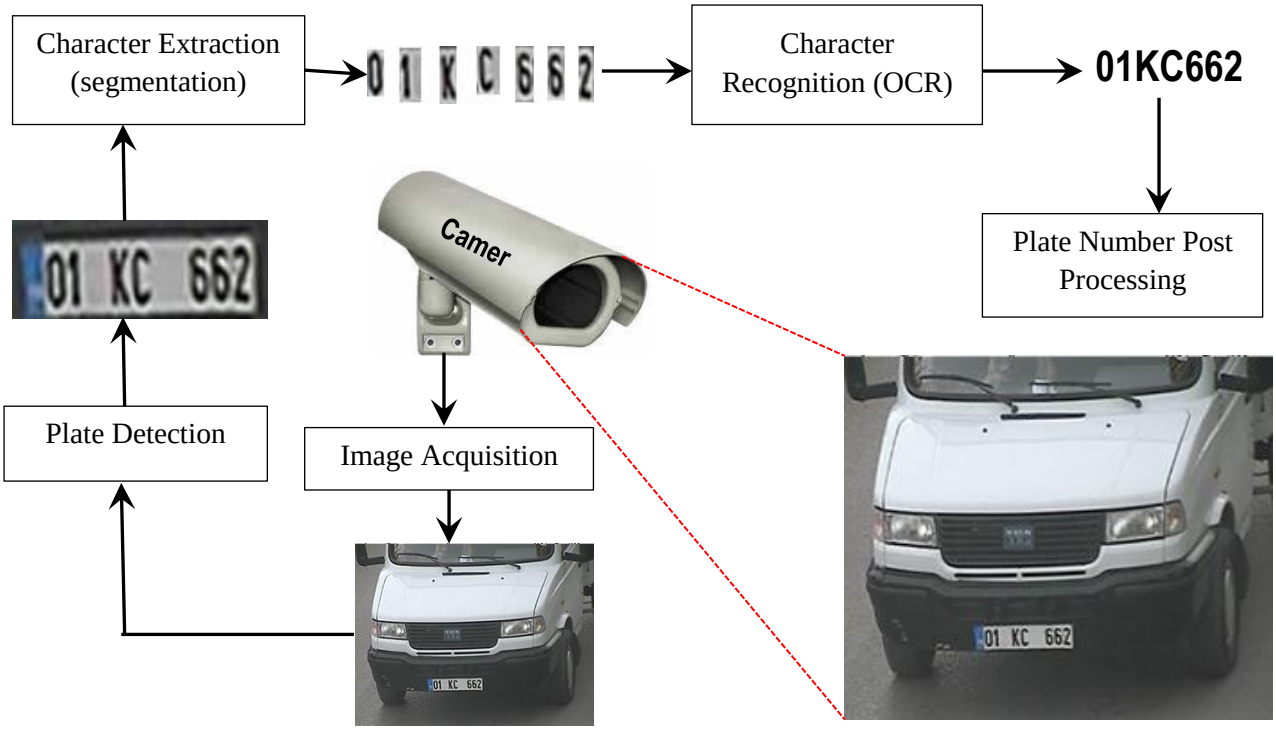


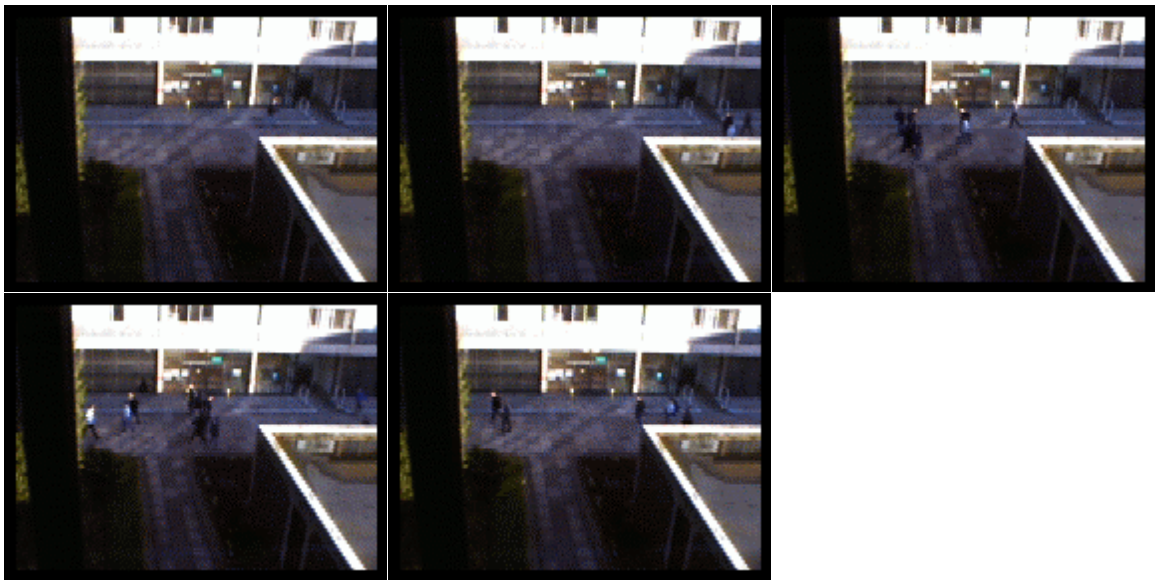
Figure 3.1. ALPR System Overview

3.1 Image Acquisition

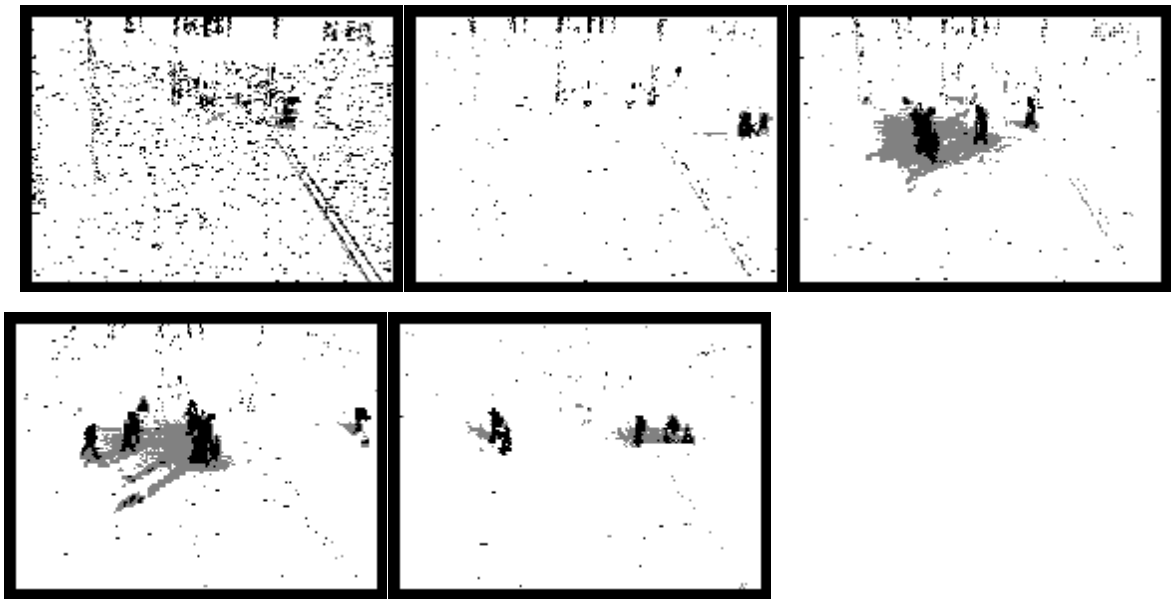
This is the only section of ALPR system that directly interact with the hardware (Camera Device), it basic role is to produce an image containing the plate number with minimal unwanted scene objects inclusion and at the fastest possible means. Processes in this part are mainly video processing techniques that provide means of generating relevant static images known as *Frames* from a continuous video sequence based on the nature of the scene. In some implementation of ALPR, this part can be further divided into two parts; the part that may be implemented in a specialised hardware commonly referred to as *Frame Grabber* that performs the direct interaction with the camera and produces frames; and the part that is fully implemented in software which perform some filtration of the frames before finally passing the filtered frame on to next stage of processing.

Typical frame filtration technique is *Background Subtraction*. Background subtraction involves calculating a reference image, subtracting each new frame from this image and thresholding the result [37 - 39]. Using this technique, when processing continuous frames sequence, a reference image is always calculated that will be subtracted from a newly

acquired frame, this gives a much smaller image that will contain only the objects in motion relative to the fixed frame or background, as such the background is subtracted from this new frames there by filtering out the unwanted non-changing or fixed frames from the video sequence.



(a)



(b)

Figure 3.2. (a) The original sequence at frames 15, 105, 235, 290 and 1200 respectively of a given video input showing the fixed and moving objects from one frame to another; (b) indicates the result of applying Background Subtraction in filtering and reducing these frames to containing only the changing or objects in motion [37].

3.2 Plate Detection

After the image acquisition stage, an image will be produced that contain potential location of vehicle plate, the process of exploring and detecting this potential location is referred to as *Plate Detection.* This is the most crucial part of ALPR system as its result greatly affects the performance of a given ALPR system. Output from this stage gives a clear cut of where specifically the plate is located, as such, if a wrong location is choosing, the ALPR system automatically fail in retrieving the plate number from the given image. A good ALPR system must have a very high Plate Detection rate, and a very low plate false detection rate. In a nut shell, success at this stage presumably means success for the ALPR system and failure automatically means failure of the ALPR system. Plate detection algorithms often define the methodology used by the ALPR system. This stage tends to take majority of the overall computation/processing time of the ALPR system as it involves image processing of a much larger image than any other subsequent stages in the ALPR system.

Table 3.1. The plate detection process of slidding concentric windows algorithm

For *each pixel image* I_1
run *SCW method in* I_1
{Parameters of SCW: X_1, Y_1, X_2, X_2, *T and M=mean value; }*
//parameters are set according the application setup. Details can be found in Table IV.
name *the resulting image* I_{AND}
perform *image masking*
calculate *image* I_2 *, where* $I_2 = I_1 \cap I_{AND}$*;*
for *each pixel in image* I_2*,*
*{****binarize*** I_2 *using Sauvola method with parameters: k=0.5, R=128, b=10;}*
name the resulting image I_3
perform *binary measurements in* I_3 *objects*
*{****retrieve*** *objects which fulfil the following measurements:*
(aspect ration > 2) and Orientation<35 degrees) and (Euler Number > 3)};
count *n, where n:number of objects detected*
If n>0 in I_3*{* $I_3 = I_4$ *and proceed to step 6;}*
If n=0 in I_3*, {invert* I_1 *(*$I_1 = - I_1$*) and* ***perform*** *steps 1,2,3,4 and 5 again;*
If n=0 in the inverted image I_1 *{print "No license plates found"}*
else { $I_3 = I_4$ *and proceed to step 6;}*
for *i=1:n image* I_4
{detect coordinates of Xmin Xmax Ymin Ymax for object I;}
Output*: matrix A(i) where A(i)= (Xmin, Xmax ,Ymin, Ymax);*

Plate detection usually involves application of image processing techniques on an image in order to separate the plate object from the image background and other of non-plate image objects. As discussed in chapter one (1), one of the notable plate detection algorithm is the sliding window technique [17]. Fig. 3.3 shows the detail pseudo code of this algorithm.

This stage is the main focus of this research; in the later chapters of this thesis, an in-depth study of plate detection process will be presented and a more efficient and accurate plate detection algorithm will be proposed.

3.3 Characters Segmentation

After the plate number location is determined, the region of the plate indicated by this location is taken as the Region of Interest (ROI) on which all the subsequent image processing will be performed. This region is expected to contain only the plate characters and some optional signs. The plate characters are segmented and extracted individually from the ROI, the process of extracting these characters is referred to as *Character Segmentation.* One of most common and simplest character segmentation techniques is the method that explores horizontal and vertical projection of pixels [40 - 42].

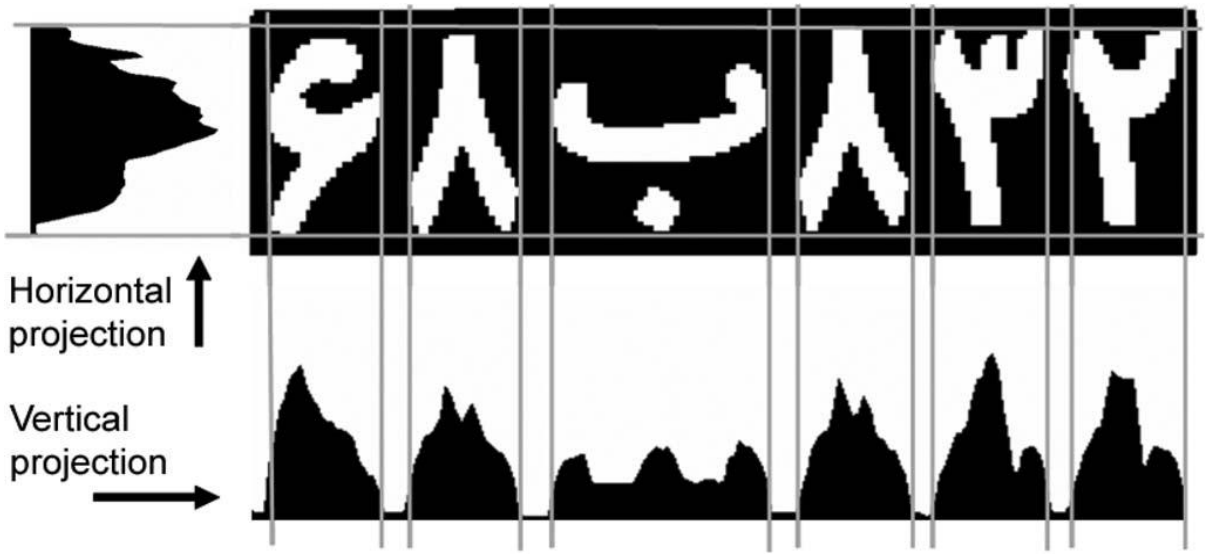


Figure 3.3. Character extraction using the horizontal and vertical projection method [11]

The method involves obtaining the sum of foreground pixels in the binary image along the vertical axis (vertical projection) as shown in Fig. 3.3. The columns where the value of the sum is minimum or zero correspond to the boundaries or spaces between the plate characters.

3.4 Character Recognition

Each of the plate characters extracted from the character segmentation stage has to be labelled as one of the known possible plate characters, the process of identifying or recognising these characters is called *Character Recognition*. The broader term used for this process is *Optical Character Recognition* (OCR). This typically involve taking each of the plate characters one at a time and passing it through a recognition module that labelled it as one of the known characters. Numerous algorithms use statistical classifiers, computational intelligence architectures, and common pattern-matching techniques. One popular algorithm used for OCR is one the of Multi-perceptron Artificial Neural Network (ANN) classifier [43 - 46].

3.5 Plate Number Post Processing

This is the final stage of an ALPR system design that distinguishes ALPR system applications. In any ALPR system application, the processes discussed above are similar. At this stage, the specific function of an ALPR system is implemented, this could be a simple function of storing the plate number into a database or even intelligently recognising the plate number to whom it belongs to.

In this chapter, general overview of ALPR systems was discussed with some specific implementation details of some of the processes in ALPR. In the next chapter, a more detail explanation of the contribution of this research will be presented with emphasis on only the relevant part of the ALPR system related to this research.

CHAPTER FOUR
PROPOSED ALGORITHM

4 Introduction

As discussed in the previous chapters, there hasn't been any specific method that is widely used in ALPR systems, each ALPR implementation introduce entirely different approach in terms of working methodology and employed algorithms. However, as already noted from the previous chapter, one thing in common to most of ALPR implementation is that they are systematically implemented in four sequential processes; image acquisition, plate number detection, plate characters extraction and finally, optical character recognition. Most of related research work described in previous chapters contributes to one or all of these processes.

This research focuses on only the plate number detection stage where an entirely new technique is proposed, that effectively detect a plate number in a complex background scene image comprising of the vehicle objects, some other background objects and the plate number object. This chapter discuss the detailed design of the proposed algorithm.

4.1 Proposed Algorithm Design

Plate detection process in ALPR system is the most crucial stage as it performs the sole role of localising the position of the plate number characters. This is why it is important to have an effective and accurate plate detection technique. An algorithm is been proposed in this research that employs three basic mechanisms to detect a plate number from a given scene image. These are:

- Structured Histogram of Oriented Gradients
- Multi-perceptron Artificial Neural Network and
- Genetic Algorithm

The proposed algorithm employs *Structured Histogram of Oriented Gradients* for the extraction of plate number image features; *Multi-perceptron Artificial Neural Network* for the classification or categorisation of a region in an image into been positive or negative plate

number region (as whether an image region is a plate or not); and finally the *Genetic Algorithm* is used to randomly choose sub-space regions of an image continuously to represent plate number regions, evaluate these regions for the possibility of presence of a plate number, until an optimised region location is obtained that will be assumed to be the plate number region.

Design details of each of these mechanisms will be expounded, followed by the overall design of the proposed algorithm in the subsequent sections.

4.1.1 Structured Histogram of Oriented Gradients

As defined in section (2.2), gradient of an image is the measure of the rate of change in the pixel values of an image along x-direction and along y-direction. Also, the gradient of an image is acquired using equation (2.2) for the gradient vector, (2.3) for the gradient magnitude and (2.4) for the gradient angle or orientation of the gradient. In this section, these gradient concepts are employed in a systematic feature extraction technique of an image region that is assumed to represents a plate number called *Structured Histogram of Oriented Gradients* (Structured HOG). It is similar to the work proposed by Dalal & Triggs (2005) [12] which fully explained in section (2.3.1). The steps for computing Structured HOG are:

- STEP 1: **Divide the Image into cells horizontally**

  The image is divided into cells horizontally to depict the structure of a typical plate number (horizontal groups of characters and spaces) as shown in Fig 4.1. Based on this, a reference plate structure is assumed, shown in Fig. 4.1, thereby making the image to be horizontally divided into 6 cells of various sizes.

- STEP 2: **Computing the Cells Histogram Vector**

  As in Dalal and Triggs proposed HOG, the gradient magnitude and orientation of each pixel in a cell division is computed. The histogram vector comprises of bins which simply represent range of orientation values from 0 to 360 or from -180 to 180 depending on whether 'signed' or 'unsigned' gradient was used. Each pixel within the cell casts a weighted vote for one of these bins based on the values found in the gradient computation. If for example, a pixel has a gradient magnitude of 10, orientation of 30 degrees and the histogram vector has 9 bins of signed gradients, then the contribution by this pixel will be at the first bin (0-40 degrees range),

depending on what is been chosen as the vote weight, if gradient magnitude is to be used, the value of the first Bin will be increased by 10.

- STEP 3: **Divide the Image into cells Vertically**

  The image is divided into cells vertically to depict the structure of rectangular nature of a plate number with border lines as shown in Fig. 4.2. Based on this, 3 vertical cell divisions of different sizes will be formed

- STEP 4: **Repeat step 2**

  Step 2 is repeated on the vertical cell divisions to form the vertical cells histogram vectors.

- STEP 5: **Cells Histogram Concatenation and Normalisation**

  Based on the assumed structure of plate image (Fig. 4.1), systematic normalization is performed on the horizontal and vertical cell divisions (*C1, C2, C3, C4, C5, C6, C7, C8 and C9).* 4 groups of normalisation were formed:

  - ✓ The cells that are assumed to contain characters are grouped together i.e. cells *C2, C4, C6 and C8.*
  - ✓ Cells that are assumed to contain lines or dark edges are also grouped. I.e. cells *C1, C7 and C9.*
  - ✓ Cells that are assumed to contain white spaces are also grouped. I.e. cells *C3 and C5.*
  - ✓ All the cells (*C1, C2, C3, C4, C5, C6, C7, C8 and C9)* are grouped together in 1 group.

  The histogram vectors of all the cells in each group are concatenated to form a single group vector, the group vector is then normalised using *L2* normalisation.

- STEP 6: **Structured HOG vector Formation**

  The normalised group vectors formed above are concatenated to form the Structured HOG vector.

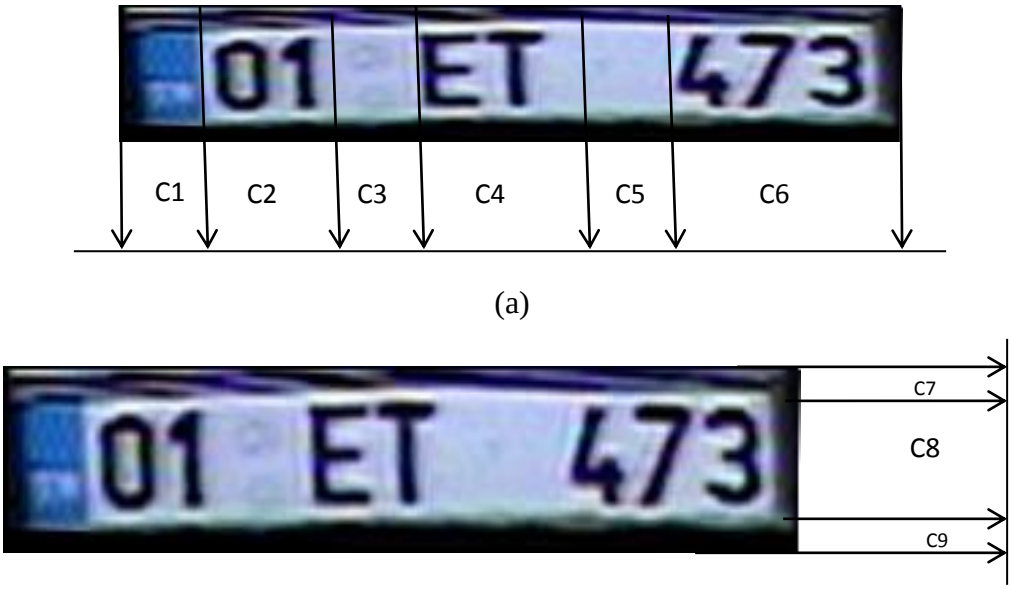


Figure 4.1. (a) and (b) assumed typical Plate structure

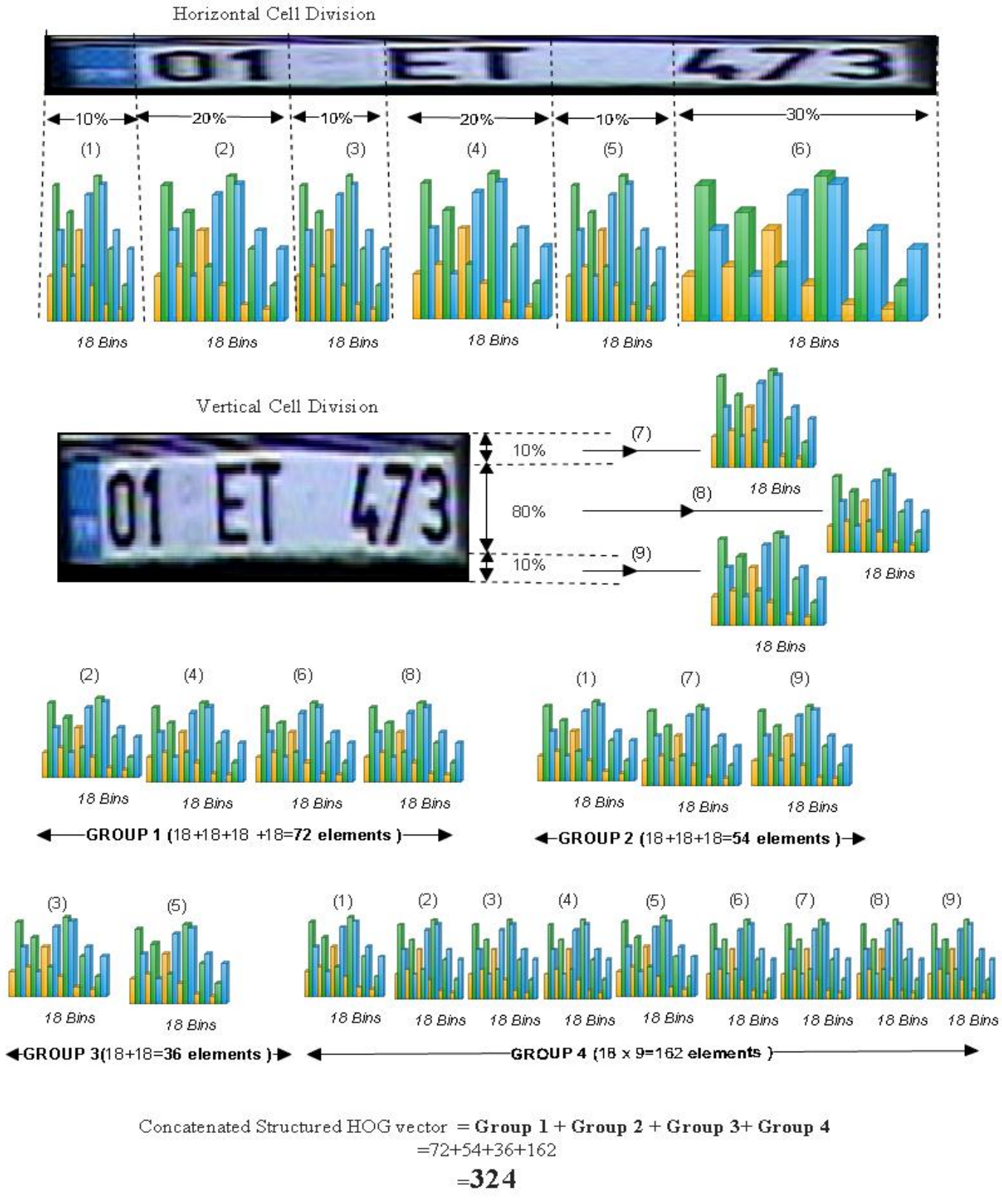
The overall Structured HOG algorithm is depicted in Fig. 4.2. It will be observed that the basic difference between the Structured HOG and Normal HOG is in the cell division and block normalisation, in the Structured HOG, the image is systematically divided based on the structure of an assumed reference plate which makes it more specific and more efficient than the normal HOG. Also the size of the Structured HOG feature vector is smaller than the HOG due to systematic cell division as opposed to the conventional cell division in the Original HOG.

An analysis of the structured HOG feature vector is presented in the next section.

4.1.1.1 Computational complexity and analysis of the proposed structured HOG

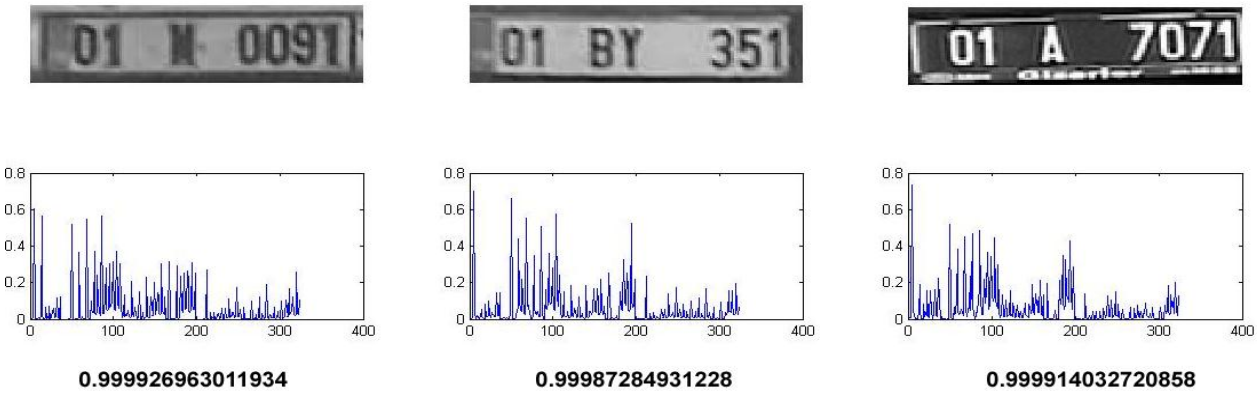
The structured HOG computation involves far less and fixed computation as the number of cells is fixed irrespective of the size of the image. The computation for example of **25** × **100** image will be equal computing histogram of a cell × **6** cells across horizontally × **3** cells across vertically=**18** cells computations and that of **2000** × **3000** image will still be = **18** cells computations. However, the size of the feature vector will be=**324** as shown in Fig. 4.2.

The Structured HOG when applied to a plate image that has printed characters with spaces produce a vector that is almost similar irrespective of the background colour and number of characters present in the plate image as shown in Fig. 4.3.

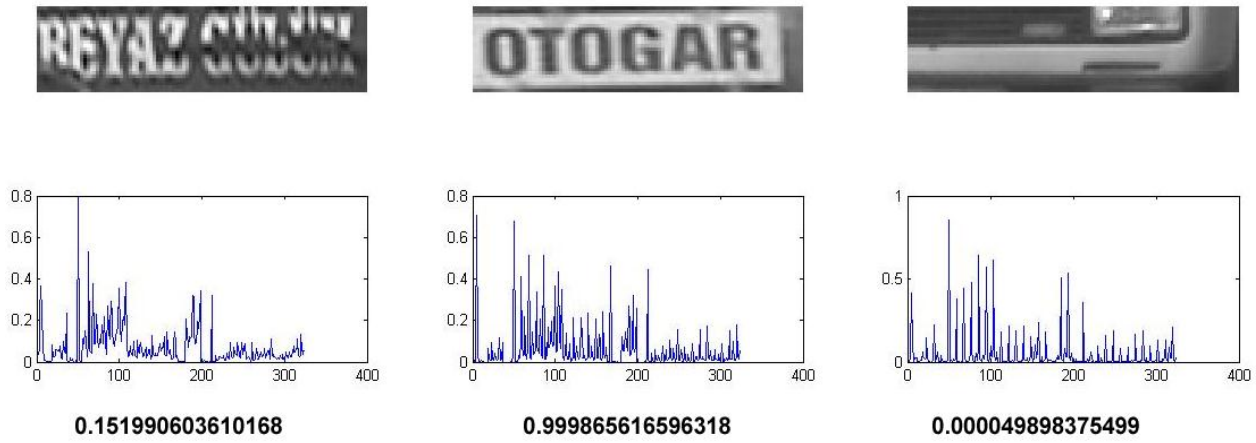


Concatenated Structured HOG vector = **Group 1 + Group 2 + Group 3+ Group 4**
=72+54+36+162
=**324**

Figure 4.2. Overview of Structured HOG algorithm



(a)



(b)

Figure 4.3. Images, Plot of Structured HOG and classification or similarity score computed with neural network classifier for (a) a sample Positive plates; and (b) Negative plates.

4.1.2 Design of Multi-perceptron Artificial Neural Network

Feed-forward multi-perceptron ANN had been discussed in detail in the previous sections (Section (2.2.1.1)). The pseudo-code of the widely used training algorithm for the ANN which is *Back-propagation* algorithm was also discussed. However, the ANN used in this research was designed to be a multilayer with only one (1) hidden layer (Fig. 4.4.). It is trained using a specialised *Back-propagation* algorithm proposed by Møller, M. F. (1993) known as *Scaled Conjugate Gradient Back-propagation Algorithm (SCG)* [47].

The *SCG* implementation is exactly based on how it is been proposed in [47]. The user defined parameters needed by this algorithm implementation are:

- Maximum number of epochs (iterations) to train
- Performance goal
- Maximum time to train in seconds
- Minimum performance gradient
- Maximum validation failures
- (*Sigma σ)* which determine change in weight for second derivative approximation
- (*lambda ʎ*) which is a parameter for regulating the indefiniteness of the Hessian

Typically one epoch of training is defined as a single presentation of all input vectors to the network. The network is then updated according to the results of all those presentations. This occurs at each iteration until any of these conditions occurs:

- ✓ The maximum number of epochs (repetitions) is reached.
- ✓ The maximum amount of time is exceeded.
- ✓ Performance is minimized to the goal.
- ✓ The performance gradient falls below certain value (minimum gradient).
- ✓ Validation performance has increased more than maximum fail times since the last time it decreased (when using validation).

Details of the Training procedures will be provided in the next chapter (chapter 5).

4.1.3 Genetic Algorithm (GA) Design

The theoretical details and a sample pseudo-code of GA had been presented in Section (2.2.2). In this work, a custom GA was designed comprising of four (4) components:

- Chromosome generation
- Fitness function
- Chromosome selection
- Evaluation function

4.1.3.1 Chromosome generation

The chromosome comprises of constrained numerical values bounded by the size of the scene image. It is divided into two genes (sections) as shown in Fig. 4.4.

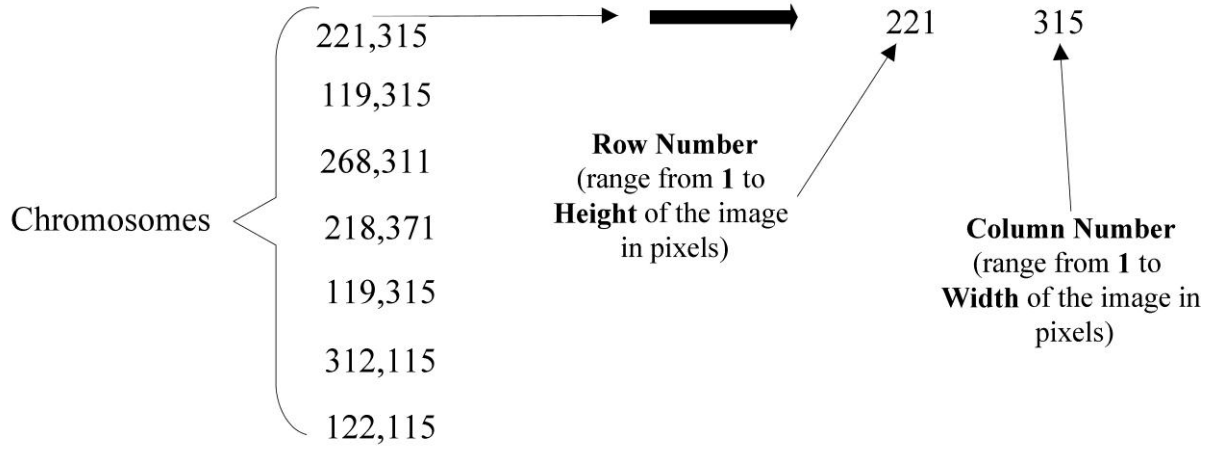


Figure 4.4. structure of chromosomes

A chromosome is represented by an array or list of two numerical values of different constraint as shown Fig. 4.4. The first gene represents the row number which is bounded by the height or total number of rows of the scene image. The second is the column number which is bounded by the width or total number of columns of the scene image.

A chromosome is formed by randomly generating a number:

- Between 1 and Total number of rows for first gene and
- Between 1 and Total number of columns for the second gene.

The two randomly generated numbers will then form the two genes for the chromosome.

4.1.3.2 Fitness Function

The fitness function grades the chromosomes and assigns a score value to a chromosome based on the classification similarity values generated by classifying that chromosome using the classifier. This can be performed in the following steps:

- STEP 1: **Extract the genes from the chromosome**

The grading or scoring process start by first extracting the genes from the chromosome which will represent the x and y values of a point in the image. The points are validated by checking the value of x which must be between 1 and width of the scene image as shown in equation (4.1). Likewise y, the values of y must lie in between 1 and height of the scene image as shown in equation (4.2).

$$x = \begin{cases} x & 1 \leq x \leq width \\ 1 & x < 1 \\ width & x > width \end{cases} \quad (4.1)$$

$$y = \begin{cases} y & 1 \leq y \leq width \\ 1 & y < 1 \\ width & y > width \end{cases} \quad (4.2)$$

- STEP 2: **Extract the sub-image**

  A rectangular region is marked using this point as the upper left vertices of this rectangle with a fixed width w and height h. w *and* h are the standard detection window size which was define globally by the GA during initialization, this rectangle is taken to be the Region of interest (ROI) on the scene image. The portion of the ROI is then extracted as the sub-image which is presumed to represent the image of the plate.

- STEP 3: **Compute the structured HOG feature vector**

  The features of the sub-image obtained in the previous step are extracted by calling the feature extraction module that performs the proposed feature extraction procedure discussed in Section (4.3.1). The module will return the structured HOG feature vector.

- STEP 4: **Finally, compute fitness value score**

  The fitness value is computed by classifying the features of the sub-image extracted above using a classifier. As described in the previous sections, feed-forward multilayer perceptron ANN was used as the classifier. The ANN output classification or similarity score is a probability value indicating the likely means of a chromosome

been a positive plate or negative plate. Normally, the output of the ANN classifier is an array of values equal to the number of the classes available, with each value representing the likely occurrence of the class and the summation of all the output values must be equal to 1. In this work, there are two classes, the positive and negative plate class. Hence, the output of the ANN classifier is a vector of two values. The value of the positive occurrence or the probability of the positive class is taken as the fitness value score. Example, let the output of the ANN= [0.3, 0.7]; and classes are [*positive, negative*]; then there is 30% chance that the classification class is *positive* and 70% chance that the classification class is *negative*, therefore, the fitness value will be 0.3. The fitness scores are between 0 and 1. The higher the fitness value of a chromosome the more likely that chromosome is considered to be a plate.

4.1.3.3 Chromosome selection

The selection of the chromosomes is the process of selecting chromosomes to be used for producing new population for the next generation or iteration during the genetic algorithm evaluation process. Chromosomes are selected based on their fitness values. In this work, *roulette selection* method was used. The detail of the algorithm is presented in Table (4.1). It can be observed that the *roulette selection* method tends to balance the trade-off between greedy selection and randomization because it equally give higher chances to chromosomes with high fitness values and also provide a possible avenue for chromosomes with low fitness values to be selected.

4.1.3.4 Evaluation Function

This is the component of the GA that is similar to the GA pseudo-code presented in Table (2.2) of Section (2.2.2). However, specific to this work, the evaluation function can be explained in the following steps:

- STEP 1: **Initialise GA parameters**
  Initialise the parameters of the genetic algorithm such as the *mutation rate, cross over rate, global maximum fitness and maximum fitness threshold.*

- STEP 2: **Generate Initial Population**

After the initialization of all the required GA parameters, an initial list of chromosomes must be created. This is the list of randomly generated location points that lie within the image co-ordinates. It simply involves a repeated call to the *Chromosome generation* function on each element of the chromosomes list.

Table 4.1. Pseudo-code of Roulette selection algorithm used in chromosome selection for the generation of a new population of chromosomes

Roulette Selection *(Fitness_Values)*

Fitness_Values: Array containing values of fitness of each chromosome.
Population : Array containing the members of the population (chromosomes)
Probability_Values: Array containing values of probabilities of each chromosome.
Cummulative_Probability_Values: Array containing cumulative sum of probabilities of each chromosome.
Sum: Total sum of the fitness values of all the chromosomes.
Cumulative_Probality_Sum: Cumulative sum of all the probabilities.
New_Population: The newly selected chromosomes.

- ***Evaluate:*** *For each p in Population,* ***compute*** $Sum = Sum + Fitness\_Values(p)$
- ***Evaluate:*** *For each p in Population,* ***compute***

$$Probability_{Values(p)} = \frac{Fitness_{Values(p)}}{Sum};$$

- ***Evaluate:*** *For each p in Population,* ***compute***

$$Cumulative_{Probality_{Sum}} = Cumulative_{Probality_{Sum}} + Probability_{Values(p)}$$

$$Cummulative_{Probability_{Values}(p)} = Cumulative_{Probality_{Sum}}$$

Create the population, *New Population:*

2. ***Do until*** *New_Population* ***is full***
   Generate a random number R between 0 and 1
   Do for each member *p* ***in*** *Cummulative_Probability_Values*
   If $R > Cummulative_{Probability_{Values}(p)}$ *AND* $R \leq Cummulative_{Probability_{Values}(p+1)}$
   Add *p* ***to*** *New_Population*

- *Return New_Population*

- STEP 3: **Evaluate Fitness Values of Chromosomes**

  The fitness values of each chromosome in the chromosomes list is computed and stored in the fitness values list by performing a repeated call to the *fitness function* on each chromosome.

- STEP 4: **Select Maximum Fitness Chromosome**

The fitness values of the chromosomes are sorted and the maximum is selected, if the maximum value is greater than the *global maximum fitness* value, then the *global maximum fitness* value is replaced with the current maximum fitness value.

- STEP 5: **Check for Termination**

  Check if the *global maximum fitness* is greater than the *maximum fitness threshold* or the iteration is greater than the *maximum iteration* then stop GA and return the chromosome with the *global maximum fitness.* If not, progress to Step 6.

- STEP 6: **Form the New Population**

  Call the roulette selection module discussed in section (4.1.3.3) to select a new population. Replace the current population with the new population which forms up a new set of chromosomes.

- STEP 7: **Perform GA Cross over operation**

  Crossover operation had already been explained in section (2.2.2.1). Since the genes in the chromosomes are only 2, only *Single-point crossover (2.2.2.1.1)* operation can be performed. Unlike the *Single-point crossover* operation explained previously that involves binary chromosomes with a *crossover mask*, the operation performed in this work is on a numerical chromosomes as such, the *Single-point crossover* operation is simply swapping of the two values in the chromosome. To perform crossover operation, the steps are.

  - ✓ Randomly select the parent chromosomes pair that will participate in the crossover operation, the number of chromosome pairs selected equals to the half of the crossover fraction of the total number of population.
  - ✓ On each pair of the chromosomes selected, randomly select either 1 or 2 which corresponds to the point of crossover or swapping between the two chromosomes in the pair. For example, let the two chromosomes in the pair be C1=043, 234 and C2=415, 215. If the number 1 is selected at random, this implies the swapping is to occur at position 1 and the newly formed chromosomes after the swapping or crossover are C3=<u>415</u>, 234 and C4=<u>043</u>, 215. The newly formed pair is the crossover sibling pair.
  - ✓ Replace each selected chromosome pair with its corresponding crossover sibling pair generated above.

- STEP 8: **Perform GA Mutation Operation**

  GA mutation operation discussed at section (2.2.2.2) involves the flipping of a bit at a randomly selected gene in a chromosome. However unlike mutation discussed at section (2.2.2.2) which is on a binary chromosome, in this work, numerical chromosome was used, as such flipping of a bit cannot be possible, however, mutation is performed by simply re-generating the gene at the specified point in the chromosome. It performed as follows:

  - ✓ Randomly select the genes to be mutated, the number of genes to be mutated equals the mutation fraction of the total number of genes in all the chromosomes (combination of all the genes in all the chromosomes are considered). This implies, generation of random numbers between 1 and total number of genes (total number of genes is the summation of all the genes in all the chromosomes).
  - ✓ On each of the selected genes above, determine the location and the chromosomes where the gene belongs to and generate a new value of that gene at that position in the chromosome.

- . STEP 9: **Repeat the Steps 3 to 8**

  Steps 3 to 8 are repeated until the GA terminates at Step **5**.

4.1.4 Overall Algorithm

The overall proposed algorithm simply involves the retrieval of the scene image and performing GA on the image to obtain the presumed plate number region as shown in the simple flowchart below.

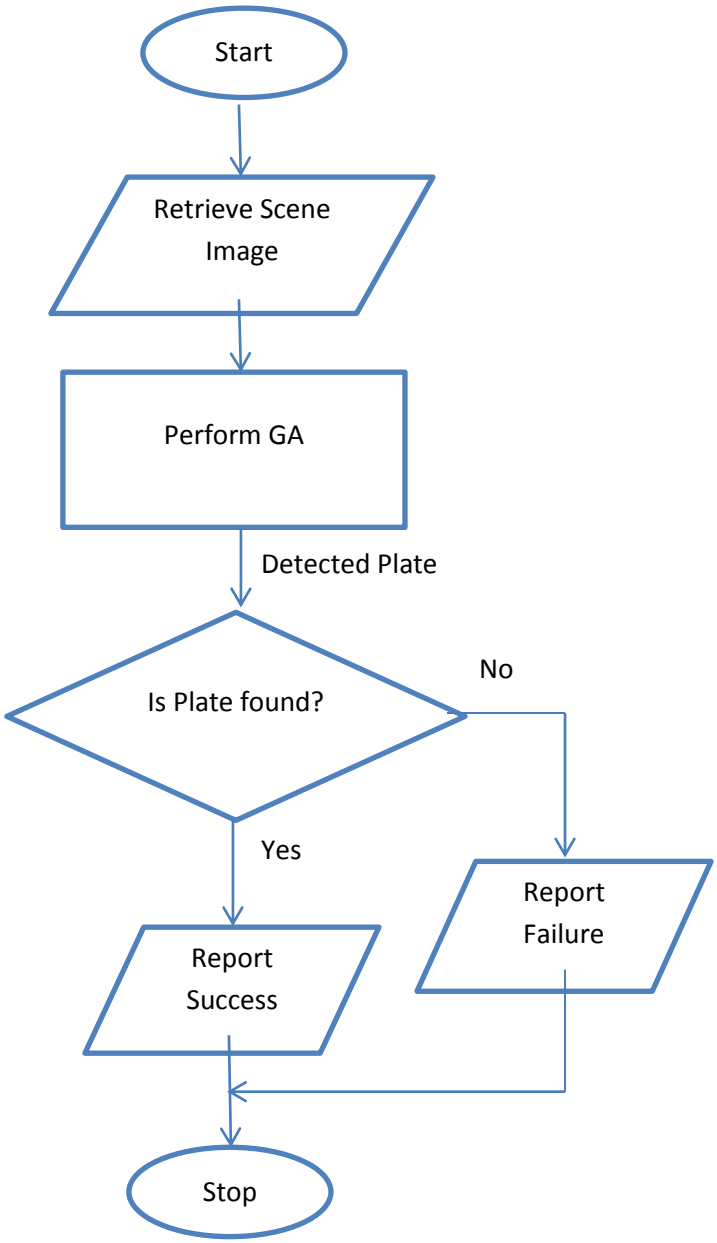


Figure 4.5. Overall Flowchart of the proposed algorithm

CHAPTER FIVE
EXPERIMENTS, RESULTS AND DISCUSSIO

5 Introduction

Theoretical explanations, methodologies and design technical details have well been discussed in the previous chapters. However, in this chapter, the experiments performed in other to implements the previously proposed algorithmic design will be presented together with results obtained as well as reasonable explanations regarding the performance of each result.

The chapter will discuss the overview of the research experimental structure with its different phases, implementation methods and result performance criteria. Details of each experiment performed, its result outcome and result discussion will be outlined.

5.1 Research implementation overview

As stated in the previous chapters, the basic aim of this research is to implement an effective ALPR plate detection or localisation algorithm which employs HOG, GA and ANN. A lot of experiments had been conducted in order to evaluate the effectiveness of the proposed plate detection algorithm in this research that will be presented in the subsequent sections of this chapter.

5.1.1 Implementation Environment and Specifications

All the experiments were performed on a desktop computer with the following specifications:

- 64 bit Windows 7 enterprise operating system
- 4 GB of RAM
- Intel® Core™ i5 -2400 CPU at 3.10GHz 3.10 GHz Processor

The experiments were implemented in MATLAB programming language and run with MATHLAB R2012a product.

5.1.2 Databases

In this research, three types of databases where used in this research:

- **Database 1** : The database which contains positive and negative license plate only
- **Database 2** : Real car scene images database (Turkish plates)
- **Database 3** : Real car scene images database (Non-Turkish plates)

5.1.2.1 Database 1: positive and negative license plate only database

This database comprises of manually cropped images of license plates (positive plates) of mainly **Turkish** license plate numbers and manually cropped images of non-license plates (negative plates). These images were extracted manually from some of the car scene images. The images are:

- **1437** Positives plates images and
- **2359** Negative plates images

All the images are **100 x 25** grey scaled images. For the purpose of demonstration, the colour version of the database is shown in this thesis. Fig. 5.1 presents some portion of the positive and Fig. 5.2 some portion of the negative plates.

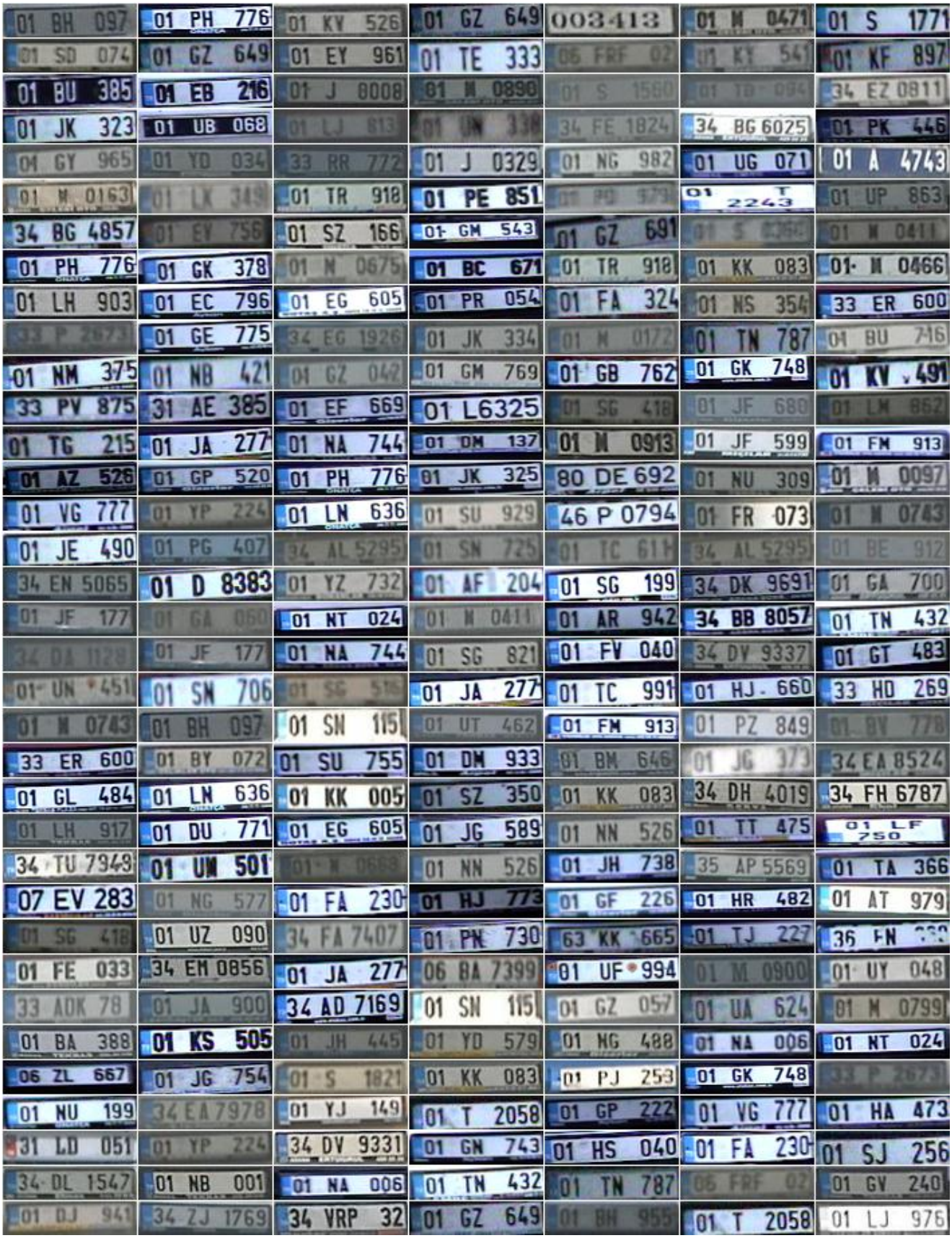


Figure 5.1. Some Colour Positive Plate images of database 1

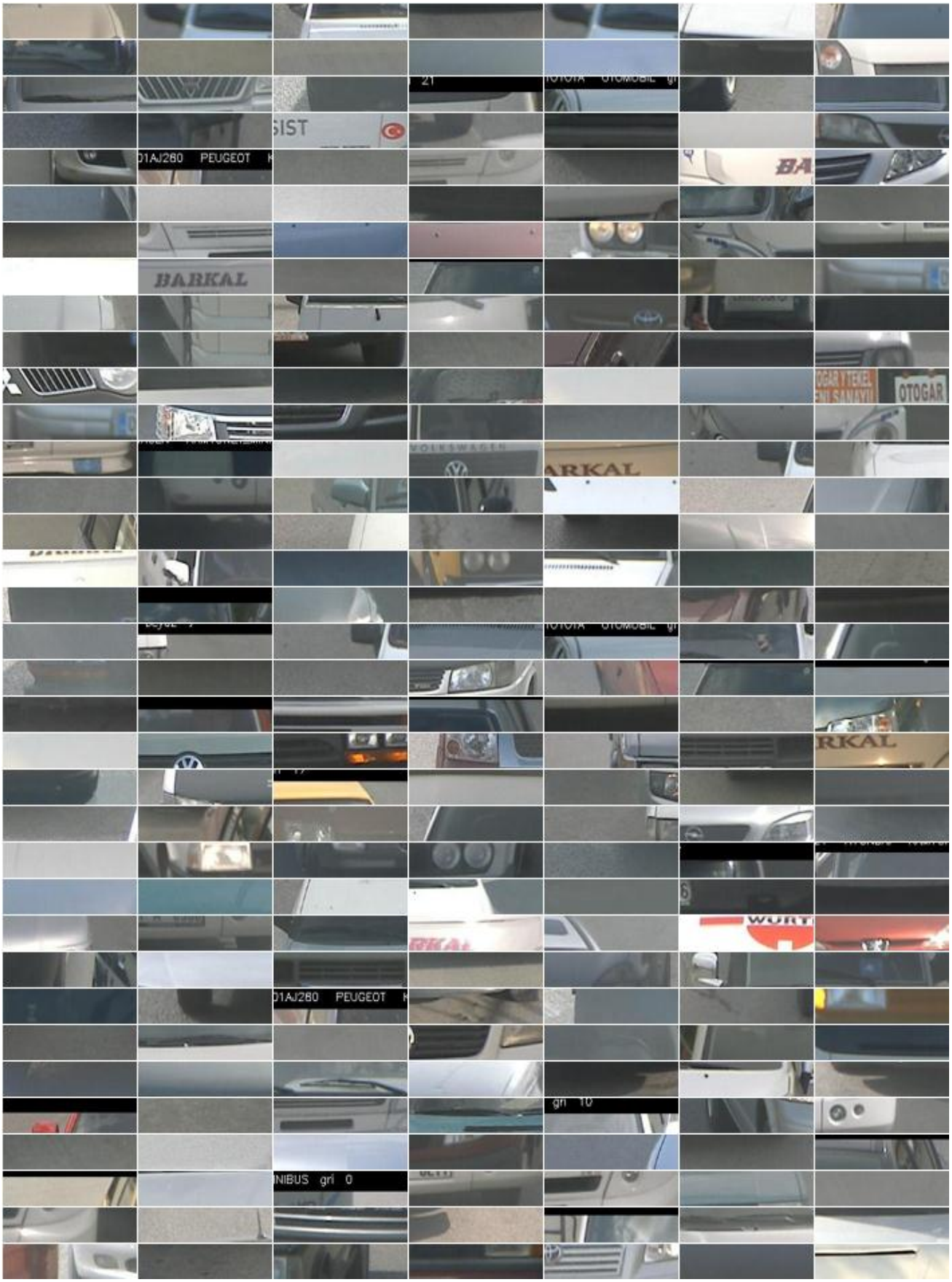


Figure 5.2. Colour Negative plate images of database 1

5.1.2.2 Database 2: Real car scene images database (Turkish)

This database comprises of scene images of cars moving on the roads at a varying distance from the camera. Pictures of the cars taken comprises of cars from various car manufacturers commonly used in turkey such as, BMC, Fiat, Ford, Ford Van, Honda, Hyundai, Iveco, Mitsubishi Van, Opel, Peugeot, Peugeot Karsan, Renault, Scoda, Tofas, Toyota, Vw Van, etc.

Each image is a **480 x 360** colour image that contains the car images with license plates predominantly Turkish plate numbers attached to these cars as shown in Fig. 5.3. The database has a total of **1572** images.

Each image has a corresponding line record in a label text file that holds all the exact positions of the license plate in the images. This information is what is use in affirming the success or failure of the plate detection process.

5.1.2.3 Database 3: Real car scene images database (Non-Turkish)

These are group of databases downloaded from a publicly available LPR database site at [48]. This database comprises of the following groups of images:

- **Group 1:** Day Blurred (**7** images of size 640 x 480)
- **Group 2:** Day Colour (**205** images of size 640 x 480)
- **Group 3:** Day Grey Scale (**48** images of size 424x 500)
- **Group 4:** Day Shadows in Plate (**49** images of size 640 x 480)
- **Group 5:** Day Very Close View (**122** images of size 640 x 480)
- **Group 6:** Difficult Dirt Shadows (**161** images of size 640 x 480)
- **Group 7:** Difficult Shadows (**26** images of size 640 x 480)
- **Group 8:** Difficult Tracks (**40** images of size 640 x 480)
- **Group 9:** Night Flash (**3** images of size 640 x 480)

License plate numbers in all the groups are mainly **Greece** license plates. Some portions of each of these groups are shown in Fig. 5.4.

Figure 5.3. Car scene image images of database 2

(a): **Group 1** -Day Blurred (**7** images of size 640 x 480) of database 3

(b): **Group 2** -Day Colour (**205** images of size 640 x 480) of database 3

(c): **Group 3-** Day Grey Scale (48 images of size 424x 500) of database 3

(d): **Group 4-** Day Shadows in Plate (**49** images of size 640 x 480) of database 3

(e): **Group 5-** Day Very Close View (**122** images of size 640 x 480) of database 3

(f): **Group 6-** Difficult Dirt Shadows (**161** images of size 640 x 480) of database 3

(g): **Group 7-** Difficult Shadows (**26** images of size 640 x 480) of database 3

(h): **Group 8-** Difficult Tracks (**40** images of size 640 x 480) of database 3

(i): **Group 9-** Night Flash (**3** images of size 640 x 480) of database 3

Figure 5.4. (a) – (i) sample images from Database 3

5.1.3 Research experiments implementation phases

With every experiment modifying or using a different set of parameters in implementing the proposed Structured HOG (Section (4.1.1)), this lead to almost every experiment having its own independently trained ANN classifier, as such experiments carried out in this research are conducted in two phases:

- **Phase 1:** Training, Testing and evaluation of the classifier i.e. ANN and
- **Phase 2:** Plate Detection on a real time car scene images.

A lot of experiments had been performed in order to evaluate the proposed plate detection algorithm. However, all the experiments performed comprises of only this two phases. As such, this section will explain the basic methodology behind these two phases.

5.1.3.1 Phase 1: Training, testing and evaluation of the ANN classifier

In this phase, *Database 1* described in section (5.1.2.1) is used in training the ANN classifier; this is explained by the following steps:

- **STEP 1**: **Extracting the feature vectors**

  The feature vector of each of the images in *Database 1* is extracted using the proposed plate number feature extraction algorithm with appropriate parameters and assembled into a single large Matrix named *Data Matrix* with each feature vector represented by a column in the *Data Matrix*. A label matrix named *Target Matrix,* that has 2 rows and a number of column equals the *Data Matrix* is also generated. Each column in the *Target Matrix* has a value 1 in its first row and 0 in its second if the corresponding feature vector is for a positive license plate and vice versa if the corresponding feature vector is for a negative license plate.

- **STEP 2: Dividing the data into training, testing, and validation set**

  The *Data Matrix* is divided into 3; Training (80%); Testing (10%); and Validation (10%) sets. This ratio is maintained in all the experiments in this research. The selection is always random and the composition of each set is different in all the experiments.

- **STEP 3: Training ANN using training set and validation set**

  The ANN training algorithm described in Section (4.1.2) is the training algorithm adopted. MATLAB 2012a implementation of neural network was the tool used in training the ANN by passing the training and validation set together with some parameters to it. Parameters used in the training are specific to each experiment and therefore will be highlighted at each of the experiments sections. The validation set is used to determine the training termination criteria.

- **STEP 4: Testing and evaluation of the ANN**

  The trained ANN classifier will then be tested using the Testing set. Classification errors are calculated and other performance goals are also measured.

- **STEP 5: Evaluating the ANN training**

  Classification errors and the performance goals are studied and evaluated, if the classification error is above a certain level, **Steps 2** to **4** are repeated until the classification error is within an acceptable level.

The steps above are performed as necessary stages of every experiment in this research. Simply, to begin with any experiment, these steps are first followed to train the ANN and subsequent stages of the experiment follow.

5.1.3.2 Phase 2: Plate detection on a real time car scene images

With a trained ANN, *Database 2* and *Database 3* described in Section (5.1.2.2) and Section (5.1.2.3) respectively are used in conjunction with GA described in Section (4.1.3) to evaluate the detection performance of the ANN. On either of the two databases (*Database 2* or *Database 3*) GA is run with each scene image in the database. The following steps explain how this is possible:

- **STEP 1: Run GA on the scene image**

  The scene image is first pass to the GA, the GA runs and returns a rectangular portion of the image that is presume to be the detected license plate.

- **STEP 2: Verify detection result**

The detected license plate obtained, is verified with the actual license plate location which is contained in a label file. The verification is simply finding the overlapping area between the GA detected rectangle and the actual rectangle retrieved from the label file, if the overlapping area is greater than zero, the detection is reported to be successful and failure otherwise.

- **STEP 3: Repeating on all the images in the database**

  Steps **1** to **2** are repeated on every image of the database selected (*Database 2* or *Database 3*), each time recording the result.

- **STEP 4: Compute detection results**

  The total number of successes and failures are counted; the overall detection performance is calculated by simply dividing the total successes with the total number of images in the database. The result obtained is then reported.

5.2 Simulations

Experiments in this research are categorised into three categories:

- Parameters Selection
- Algorithm performance evaluation
- Plate detection test running on non-Turkish plate database (Database 3)

As these experiments involve changes in the feature extraction algorithm with the exception of the third category, the GA and ANN training parameters are fixed in all the experiments. Values of the GA parameters used in all of these experiments are:

- *Maximum iteration* =75
- *No of population* =20
- *Mutation rate* =0.8
- *Cross over rate* =0.8
- *Max fitness value* =0.999;
- *Detection window size* = 30 x 130

And ANN training parameters values are:

- *No. of neurons* =15
- *Training algorithm* = Scaled conjugate gradient backpropagation
- *Maximum number of epochs to train* = 100
- *Maximum time to train in seconds* = Infinity
- Minimum performance gradient=1 e-6
- *Maximum validation failures* =5
- *change in weight for second derivative approximation* = 5.0e-5
- *Indefiniteness of the Hessian* = 5.0e-7

5.2.1 Parameters selection experiments

This category involves experiments that are aimed at fine tuning the parameters used by the proposed plate detection algorithm which are suitable parameters for its efficient implementation. For the proposed algorithm, it can be observed that there are some parameters that directly affect the performance of the algorithm, these are:

- Number of bins in a histogram. Specific values used are: *5*, *9*, *18*, *27* and *36*.
- Enhancement strategy (Smoothening with either *3 x 3 Gaussian*, *5 x 5 Gaussian* or *No smoothening*)
- Addition of statistical features by concatenating of either *variance* of a cell, *edge density* of a cell or *both* to the cell histogram vector.
- Voting strategy in bins formation of the Structured HOG by either voting with the conventional *gradient magnitude* or using only the absolute value of the *vertical gradient*.

Experiments are carried out to determine the optimal values of each of these parameters, as such; the details of the experiments will be discussed in the subsequent sections.

5.2.1.1 Experiments to determine number of bins in a histogram

These are experiments that are carried out in order to determine the optimal number of bins in the histogram of the structured HOG. The parameters fixed in these experiments are:

- *Enhancement strategy* = NO Smoothening.
- *Bin Voting* = Gradient magnitude
- *Statistics* =None

5 experiments where performed, with number of bins *5, 9, 18, 27* and *36* used respectively.

5.2.1.1.1 Experiment 5.1: 5 bins Structured HOG

ANN training phase:

Table 5.1. 5 bins experiment Confusion matrix (in %) for Testing dataset (380 images)

	Positive	**Negative**	**Total**
Positive	100	0	
Negative	0	100	
Total			**100**

Table 5.2. 5 bins experiment Combined confusion matrix (in %) for whole dataset (3796 images)

	Positive	**Negative**	**Total**
Positive	99.79	0.21	0
Negative	0.21	99.79	0
Total	0	0	**99.79**

Scene car images testing phase:

Table 5.3. 5 bins experiment Results of running GA 5 times on Database 2

Result of 5 runs (%)					
1	**2**	**3**	**4**	**5**	**Average**
89.12	89.44	90.52	89.69	89.95	**89.75**

5.2.1.1.2 Experiment 5.2: 9 bins Structured HOG

ANN training phase:

Table 5.4. 9 bins experiment Confusion matrix (in %) for Testing dataset (380 images)

	Positive	Negative	Total
Positive	100	0	
Negative	0	100	
Total			**100**

Table 5.5. 9 bins experiment Combined confusion matrix (in %) for whole dataset (3796 images)

	Positive	Negative	Total
Positive	99.93	0.07	
Negative	0.13	99.87	
Total			**99.89**

Scene car images testing phase:

Table 5.6. 9 bins experimental results of running GA 5 times on Database 2

Result of 5 runs (%)					
1	**2**	**3**	**4**	**5**	**Average**
93.19	92.49	93.38	93.19	92.94	**93.04**

5.2.1.1.3 Experiment 5.3: 18 bins Structured HOG

ANN training phase:

Table 5.7. 18 bins experiment Confusion matrix (in %) for Testing dataset (380 images)

	Positive	Negative	Total
Positive	100	0	
Negative	0	100	
Total			**100**

Table 5.8. 18 bins experiment Combined confusion matrix (in %) for whole dataset (3796 images)

	Positive	Negative	Total
Positive	99.79	0.21	
Negative	0.08	99.92	
Total			**99.87**

Scene car images testing phase:

Table 5.9. 18 bins experimental results of running GA 5 times on Database 2

Result of 5 runs (%)					
1	**2**	**3**	**4**	**5**	**Average**
95.17	94.97	95.23	94.72	95.04	**95.03**

5.2.1.1.4 Experiment 5.4: 27 bins Structured HOG

ANN training phase:

Table 5.10. 27 bins experiment Confusion matrix (in %) for Testing dataset (380 images)

	Positive	**Negative**	**Total**
Positive	100	0	
Negative	0	100	
Total			**100**

Table 5.11. 27 bins experiment Combined confusion matrix (in %) for whole dataset (3796 images)

	Positive	**Negative**	**Total**
Positive	99.93	0.07	
Negative	0.04	99.96	
Total			**99.95**

Scene car images testing phase:

Table 5.12. 27 bins experimental results of running GA 5 times on Database 2

Result of 5 runs (%)					
1	**2**	**3**	**4**	**5**	**Average**
95.67	96.88	96.06	96.56	95.93	**96.22**

5.2.1.1.5 Experiment 5.5: 36 bins Structured HOG

ANN training phase:

Table 5.13. 36 bins experiment Confusion matrix (in %) for Testing dataset (380 images)

	Positive	Negative	Total
Positive	99.33	0.67	
Negative	0.43	99.57	
Total			**99.48**

Table 5.14. 36 bins experiment Combined confusion matrix (in %) for whole dataset (3796 images)

	Positive	Negative	Total
Positive	99.86	0.14	
Negative	0.13	99.87	
Total			**99.87**

Scene car images testing phase:

Table 5.15. 36 bins experimental results of running GA 5 times on Database 2

Result of 5 runs (%)					
1	**2**	**3**	**4**	**5**	**Average**
91.03	89.82	89.95	90.52	89.89	**90.24**

5.2.1.1.6 Discussions

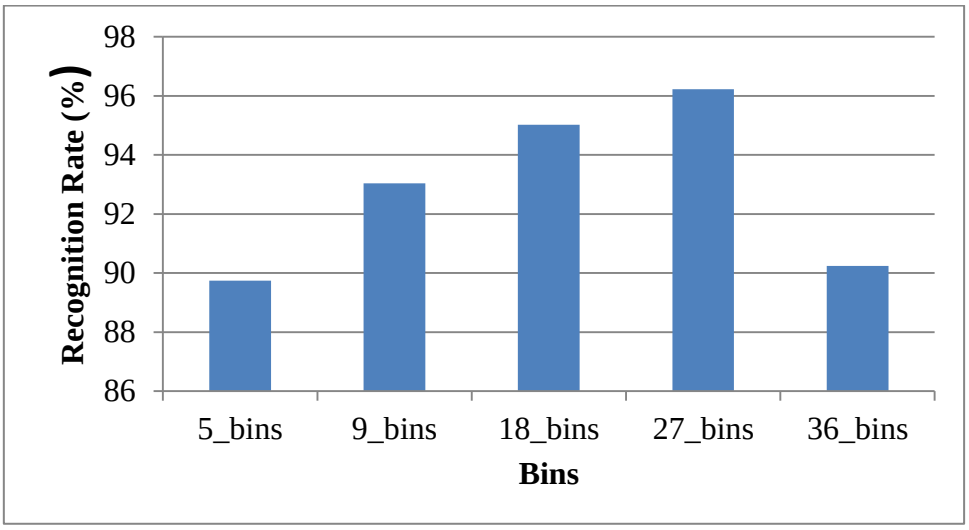


Figure 5.5. Comparison of the effect of different number of bins in our proposed structured HOG algorithm

It can be observe from the chat in Fig. 5.5 above that using structured HOG higher number of bins improves result, with **27** bins yielding the best results. However, as the number of bins increases from **27** to **36,** the recognition reduces, and this can be attributed to the texture of a plate which is relatively simple and requires less detailed bins representation because with detailed bin representation, noise could be equally emphasized and captured by the higher bins representation.

5.2.1.2 Experiments to determine enhancement strategy

These are experiments performed in order to study the effect of smoothing a plate image before classification. Two experiments where performed with 2 Gaussian filters (3x3 and 5x5 filters). The parameters fixed in these experiments are:

- *Number of bins* = 18.
- *Bin Voting* = Gradient magnitude
- *Statistics* =None

5.2.1.2.1 Experiment 5.6: 3 x 3 Gaussian Filter smoothening before computing Structured HOG

ANN training phase:

Table 5.16. 3 x 3 Gaussian Filter smoothening experiment Confusion matrix (in %) for Testing dataset (380 images)

	Positive	**Negative**	**Total**
Positive	100	0	
Negative	0.41	99.59	
Total			**99.74**

Table 5.17. 3 x 3 Gaussian Filter smoothening experiment Combined confusion matrix (in %) for whole dataset (3796 images)

	Positive	**Negative**	**Total**
Positive	99.72	0.28	
Negative	0.04	99.96	
Total			**99.87**

Scene car images testing phase:

Table 5.18. 3 x 3 Gaussian Filter smoothening experimental results of running GA 5 times on Database 2

Result of 5 runs (%)					
1	**2**	**3**	**4**	**5**	**Average**
96.06	95.99	96.25	95.99	96.12	**96.08**

5.2.1.2.2 Experiment 5.7: 5 x 5 Gaussian Filter smoothening before computing Structured HOG

ANN training phase:

Table 5.19. 5 x 5 Gaussian Filter smoothening experiment Confusion matrix (in %) for Testing dataset (380 images)

	Positive	**Negative**	**Total**
Positive	100	0	
Negative	0	100	
Total			**100**

Table 5.20. 5 x 5 Gaussian Filter smoothening experiment Combined confusion matrix (in %) for whole dataset (3796 images)

	Positive	**Negative**	**Total**
Positive	99.80	0.21	
Negative	0.04	99.96	
Total			**99.90**

Scene car images testing phase:

Table 5.21. 5 x 5 Gaussian Filter smoothening experimental results of running GA 5 times on Database 2

Result of 5 runs (%)					
1	**2**	**3**	**4**	**5**	**Average**
93.45	93.77	93.90	93.70	93.13	**93.59**

5.2.1.2.3 Discussions

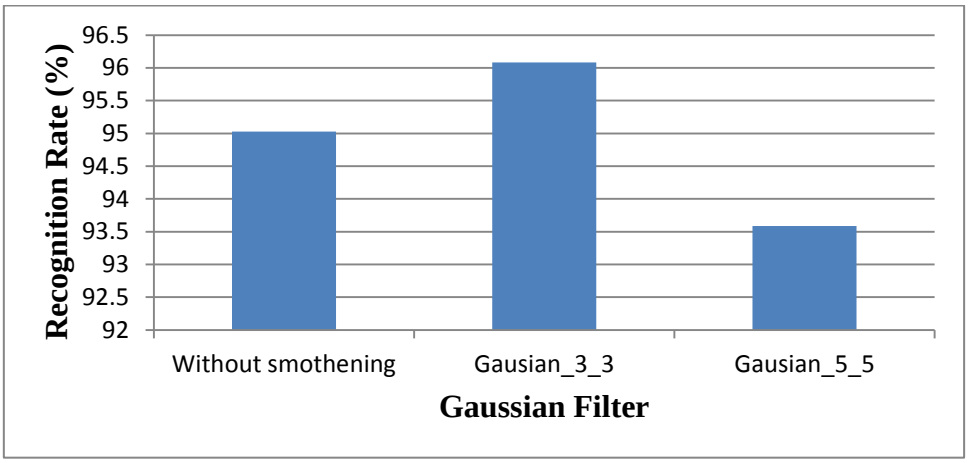


Figure 5.6. Comparison of the effect of result without applying Gaussian filter (result from experiment 5.3) and Results of applying 3x3 and 5x5 Gaussian filters in our proposed structured HOG algorithm.

It can be observed from Fig. 5.6 that by applying 3x3 Gaussian filter to the image before the computation of our proposed structured HOG, the result improves. This is evident from the fact that the presence of noise in a scene image is inevitable. However, this result depend on the nature of the images captured from the scene and the type of camera used, this is because of the dynamic environmental conditions and image quality variation across cameras.

5.2.1.3 Experiments to determine the effect of additional statistical features to the structured HOG histogram vector

These are experiments aimed at discovering the effect of concatenating an additional statistical feature of a cell to the cell histogram vector of structured HOG. The parameters fixed in these experiments are:

- *Enhancement strategy* = NO Smoothening.
- *Number of bins* = 18.
- *Bin Voting* = Gradient magnitude

Statistical features used are *variance* and *edge density* computed using equation 5.1 and equation 5.2 respectively.

Variance=$\boldsymbol{\sigma} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})^2$ (5.1)

Where $\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$

$$\boldsymbol{Edge\ density} = \frac{\boldsymbol{Number\ of\ vertical\ edge\ pixels}}{\boldsymbol{Total\ number\ of\ pixels}} \quad (5.2)$$

Three experiments were conducted by adding variance only, edge density only and combination of both to the histogram vector.

5.2.1.3.1 Experiment 5.8: Adding Edge density to Structured HOG

ANN training phase:

Table 5.22. Adding Edge density experiment Confusion matrix (in %) for Testing dataset (380 images)

	Positive	**Negative**	**Total**
Positive	100	0	
Negative	0	100	
Total			**100**

Table 5.23. Adding Edge density experiment Combined confusion matrix (in %) for whole dataset (3796 images)

	Positive	**Negative**	**Total**
Positive	99.86	0.14	
Negative	0.08	99.92	
Total			**99.90**

Scene car images testing phase:

Table 5.24. Adding Edge density experimental results of running GA 5 times on Database 2

Result of 5 runs (%)					
1	**2**	**3**	**4**	**5**	**Average**
95.23	95.23	95.48	95.17	94.59	**95.14**

5.2.1.3.2 Experiment 5.9: Adding Variance to Structured HOG

ANN training phase:

Table 5.25. Adding Variance experiment Confusion matrix (in %) for Testing dataset (380 images)

	Positive	Negative	Total
Positive	100	0	
Negative	0	100	
Total			**100**

Table 5.26. Adding Variance experiment Combined confusion matrix (in %) for whole dataset (3796 images)

	Positive	Negative	Total
Positive	99.93	0.07	
Negative	0.17	99.83	
Total			**99.87**

Scene car images testing phase:

Table 5.27. Adding Variance experimental results of running GA 5 times on Database 2

Result of 5 runs (%)					
1	**2**	**3**	**4**	**5**	**Average**
96.44	96.06	96.25	95.99	95.61	**96.07**

5.2.1.3.3 Experiment 5.10: Adding Variance and Edge density to Structured HOG

ANN training phase:

Table 5.28. Adding Variance and Edge density experiment Confusion matrix (in %) for Testing dataset (380 images)

	Positive	Negative	Total
Positive	100	0	
Negative	0	100	
Total			**100**

Table 5.29. Adding Variance and Edge density experiment Combined confusion matrix (in %) for whole dataset (3796 images)

	Positive	Negative	Total
Positive	99.93	0.07	
Negative	0	100	
Total			**99.97**

Scene car images testing phase:

Table 5.30. Adding Variance and Edge density experimental results of running GA 5 times on Database 2

Result of 5 runs (%)					
1	**2**	**3**	**4**	**5**	**Average**
96.82	96.82	97.14	97.20	97.20	**97.04**

5.2.1.3.4 Discussions

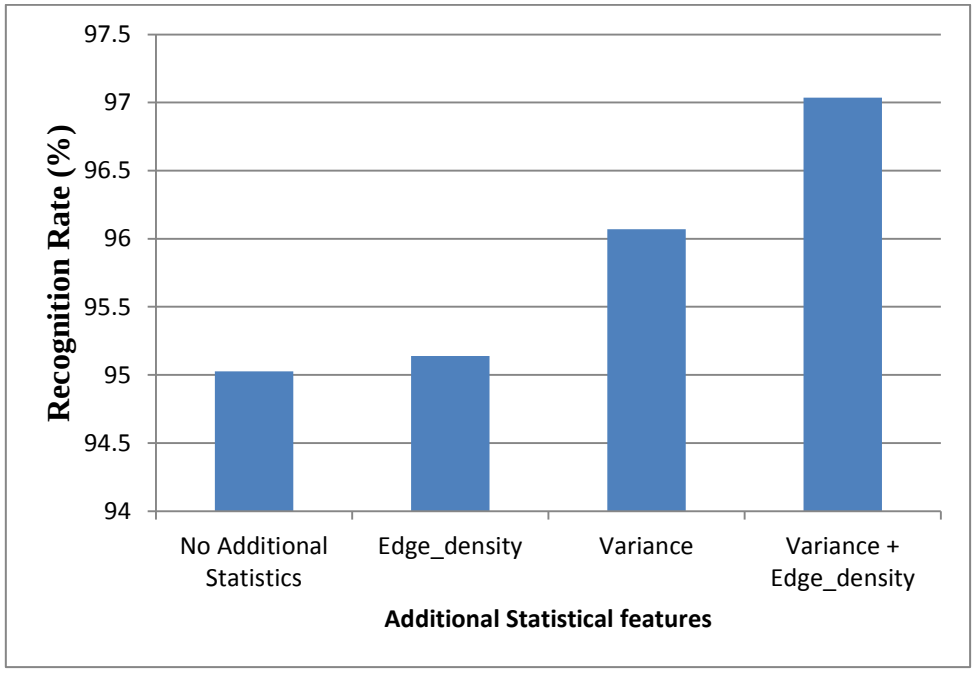


Figure 5.7. Comparison of the effect of concatenating statistical features to the histogram vector in our proposed structured HOG.

It can be observed that by adding the statistical property, the recognition rates increases, however, this can be proven by considering the fact that in the areas of plate region, there is high variation of intensity values due to the presence of textual characters, as such providing more distinctive feature to the structured HOG. Also, adding both the variance and edge density will add a more distinction, hence high recognition rates.

5.2.1.4 Experiments to determine Voting strategy in histogram bins formation of the structured HOG.

Due to the fact that the vertical gradient of a textual image is more pronounced than a non-textual image, employing this advantage in the structured HOG computation could improve performance. In this category of experiments, instead of the conventional voting strategy in histogram bin formation which is using the gradient magnitude, the absolute value of the vertical gradient was used. Specifics about the experiment will be presented in the next section. The parameters fixed in these experiments are:

- *Enhancement strategy* = NO Smoothening.
- *Number of bins* = 18.
- *Statistics* =None

5.2.1.4.1 Experiment 5.11: Vertical gradient only bin voting in histogram bins formation of the Structured HOG

ANN training phase:

Table 5.31. Vertical gradient only bin voting experiment Confusion matrix (in %) for Testing dataset (380 images)

	Positive	**Negative**	**Total**
Positive	100	0	
Negative	0.42	99.58	
Total			**99.74**

Table 5.32. Vertical gradient only bin voting experiment Combined confusion matrix (in %) for whole dataset (3796 images)

	Positive	**Negative**	**Total**
Positive	99.93	0.07	
Negative	0.34	99.66	
Total			**99.76**

Scene car images testing phase:

Table 5.33. Vertical gradient only bin voting experimental results of running GA 5 times on Database 2

Result of 5 runs (%)					
1	**2**	**3**	**4**	**5**	**Average**
83.72	84.16	84.48	83.40	82.95	**83.74**

5.2.1.4.2 Discussions

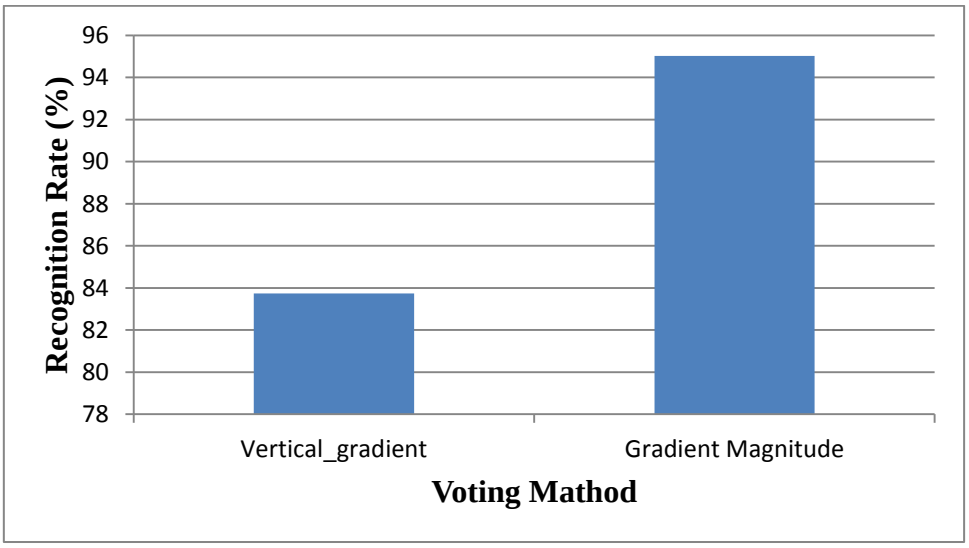


Figure 5.8. Comparison of the performance of different voting method in computing the structured HOG

Form Fig. 5.8 above, it can be observed that the prediction of using only the vertical gradient in bin voting during the structured HOG computation has proven to be wrong. However, it could be concluded that the contribution of both the vertical and horizontal gradient gives much better results. Furthermore, it has been observed that this technique could be used in detecting general textual based image region as most of the failures reported from this technique are based on the presence of textual content in the image, of which it falsely detect any text as plate.

5.2.1.5 Experiments to determine the best set of parameters

After performing the experiments above, the parameters values that produce the best result are selected and combined in order to produce a single set of best performing parameters. However, correlation was observed in combining these parameters best values as can be seen in the experiments explained in the next section.

5.2.1.5.1 Experiment 5.12: Experiment with the best values of parameters

The experiment was performed with the following parameters which produced individually the best performance:

- *Enhancement strategy* = 3 x 3 Gaussian filter.
- *Number of bins* = 27.
- *Statistics* =Edge + Variance
- *Bin Voting* = Gradient magnitude

ANN training phase:

Table 5.34. Experiments with best parameters experiment Confusion matrix (in %) for Testing dataset (380 images)

	Positive	**Negative**	**Total**
Positive	100	0	
Negative	0	100	
Total			**100**

Table 5.35. Experiments with best parameters experiment Combined confusion matrix (in %) for whole dataset (3796 images)

	Positive	**Negative**	**Total**
Positive	99.93	0.07	
Negative	0.08	99.92	
Total			**99.92**

Scene car images testing phase:

Table 5.36. Experiments with best parameters experimental results of running GA 5 times on Database 2

Result of 5 runs (%)					
1	**2**	**3**	**4**	**5**	**Average**
96.25	96.12	96.12	96.00	95.55	**96.01**

5.2.1.5.2 Experiment 5.13: Experiment with 27 bins and variance with edge statistics in the feature vector of structured HOG

The experiment was performed with the following parameters which produced individually the best performance:

- *Enhancement strategy* = none.
- *Number of bins* = 27.
- *Statistics* =Edge + Variance
- *Bin Voting* = Gradient magnitude

ANN training phase:

Table 5.37. Confusion matrix (in %) for the experiments with the best parameter values Testing dataset (380 images)

	Positive	**Negative**	**Total**
Positive	100	0	
Negative	0.42	99.58	
Total			**99.74**

Table 5.38. Combined confusion matrix (in %) for the experiments with the best parameter values whole dataset (3796 images)

	Positive	**Negative**	**Total**
Positive	100	0	
Negative	0.04	99.96	
Total			**99.97**

Scene car images testing phase:

Table 5.39. Experiments with 27 bins and variance with edge statistics experimental results of running GA 5 times on Database 2

Result of 5 runs (%)					
1	**2**	**3**	**4**	**5**	**Average**
96.18	96.95	96.95	96.50	96.82	**96.68**

5.2.1.5.3 Experiment 5.14: Experiment with 27 bins and 3x3 Gaussian filtering applied before computation of structured HOG

The experiment was performed with the following parameters which produced individually the best performance:

- *Enhancement strategy* = 3 x 3 Gaussian filter.
- *Number of bins* = 27.
- *Statistics* =none
- *Bin Voting* = Gradient magnitude

ANN training phase:

Table 5.40. Experiments with 27 bins and 3x3 Gaussian filter Confusion matrix (in %) for Testing dataset (380 images)

	Positive	**Negative**	**Total**
Positive	100	0	
Negative	0	100	
Total			**100**

Table 5.41. Experiments with 27 bins and 3x3 Gaussian filter Combined confusion matrix (in %) for whole dataset (3796 images)

	Positive	**Negative**	**Total**
Positive	99.93	0.07	
Negative	0.04	99.96	
Total			**99.95**

Scene car images testing phase:

Table 5.42. Experiments with 27 bins and 3x3 Gaussian filter experimental results of running GA 5 times on Database 2

Result of 5 runs (%)					
1	**2**	**3**	**4**	**5**	**Average**
95.17	94.91	95.04	94.97	95.17	**95.05**

5.2.1.5.4 Experiment 5.15: Experiment with variance with edge statistics and 3x3 Gaussian filtering applied before computation of structured HOG

The experiment was performed with the following parameters which produced individually the best performance:

- *Enhancement strategy* = 3 x 3 Gaussian filter.
- *Number of bins* = 18.
- *Statistics* = Edge + Variance
- *Bin Voting* = Gradient magnitude

ANN training phase:

Table 5.43. Experiments with variance with edge statistics and 3x3 Gaussian filter Confusion matrix (in %) for Testing dataset (380 images)

	Positive	**Negative**	**Total**
Positive	100	0	
Negative	0	100	
Total			**100**

Table 5.44. Experiments with variance with edge statistics and 3x3 Gaussian filter Combined confusion matrix (in %) for whole dataset (3796 images)

	Positive	**Negative**	**Total**
Positive	99.93	0.07	
Negative	0	100	
Total			99.97

Scene car images testing phase:

Table 5.45. Experiments with variance with edge statistics and 3x3 Gaussian filter experimental results of running GA 5 times on Database 2

Result of 5 runs (%)					
1	**2**	**3**	**4**	**5**	**Average**
96.63	95.87	95.87	96.06	96.76	**96.23**

5.2.1.5.5 Discussions

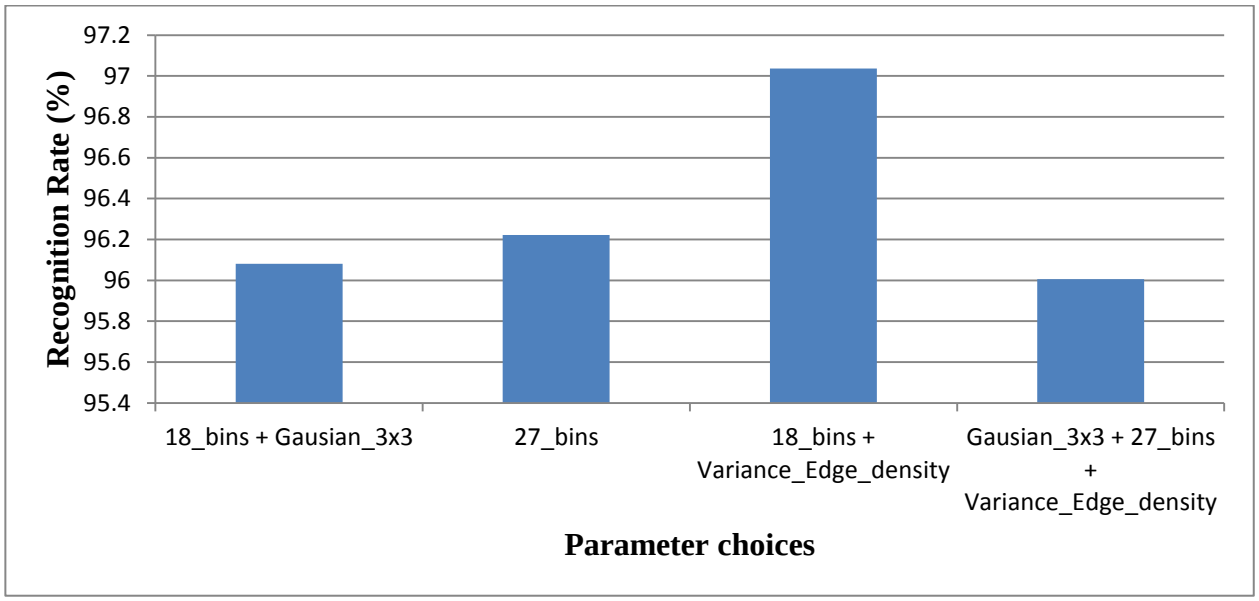


Figure 5.9. Comparison of the individual performance result of best performed parameter choices (27 bin, 3x3 Gaussian and Variance edge statistics) and the performance result of combination of all the three parameter choices.

On performing Experiment 5.12, it is evidently expected to get the most preferred performance result; however, the reverse is the case. This is because the three parameter choices when combined together will logically have a reduced performance as they are indirectly correlated. To explain this, by carefully observing Fig. 5.9 above, where the parameter choices are combined two at a time in different experiments, the performance effect of each parameter can be deduced. The following cases explain these reasons behind these effects:

- **CASE 1: 27 bin vs. Variance and Edge statistics (Fig. 5.10)**

  As proven in the previous experiments, on increasing the number of bins to 27, the performance increases, so also on adding variance and edge statistics, the performance also increases, but when these two parameter choices are combined, the performance is also expected to increase or remain constant. However, result obtained indicates a slight reduction in performance. This can only be attributed to the randomization error of the GA during the detection process.

- **CASE 2: 27 vs. 3x3 Gaussian filter (Fig. 5.11)**

By increasing the number of bins to 27 and at the same time applying Gaussian filter, the performance is logically expected to reduce, this is because as more bins are used, more detailed features is captured and hence more performance, but with Gaussian filtration, the smoothening tends to supress some important details which can cause the larger bins to be little bit redundant and hence will equally lead to misclassification errors. The result obtained (**95.05%**) is less than results obtained by using the two parameter choices individually (3x3 Gaussian has **96.08%** and 27 bins has **96.03**%).

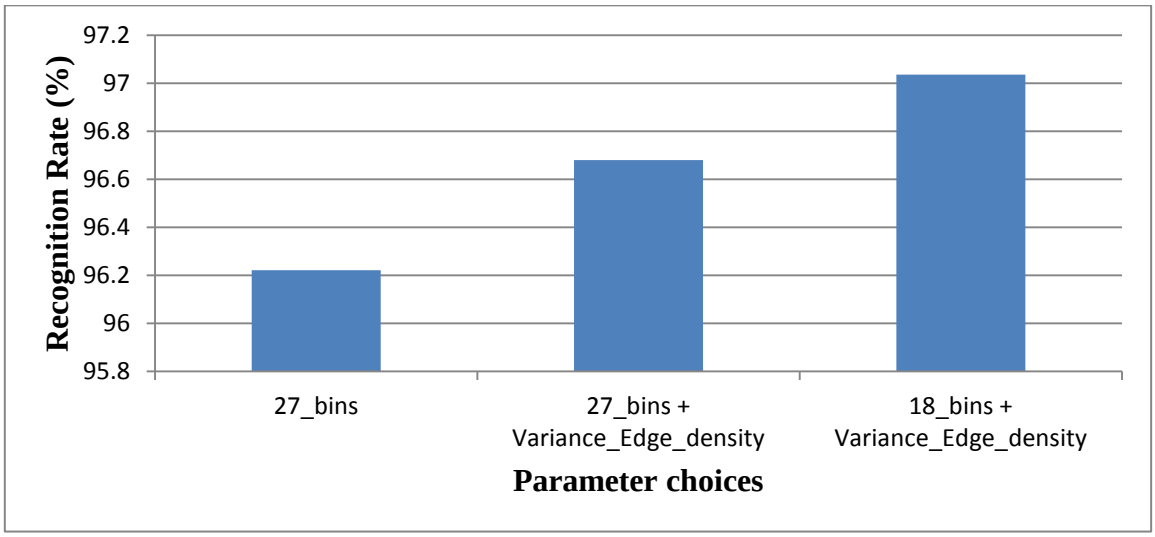


Figure 5.10. Comparison of the individual performance result of the 27 bin and Variance with edge statistics parameter choices and the performance result of combination of the two

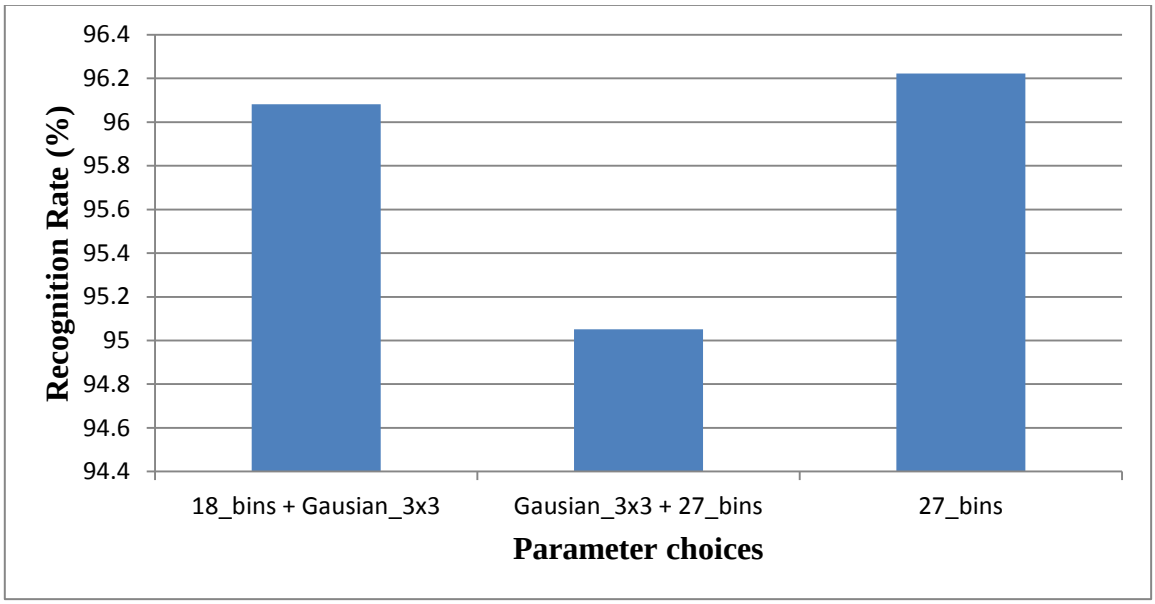


Figure 5.11. Comparison of the individual performance result of the 3x3 Gaussian and 27 bins parameter choices and the performance result of combination of the two

- **CASE 3: 3x3 Gaussian filter vs. Variance and Edge statistics (Fig. 5.12)**

  This combination is pretty straight forward, Gaussian smoothening leads to lesser variation in intensity values hence can results to reduced variance and smoothened edges, this can of course causes the computed statistical features to be less distinctive which will then reduce the overall detection performance. Result obtained is improved (**96.23 %**) when compared with using only the 3x3 Gaussian (**96.08%**) parameter choices but reduces when compared with using only the variance and edge density features alone (**97.04%**).

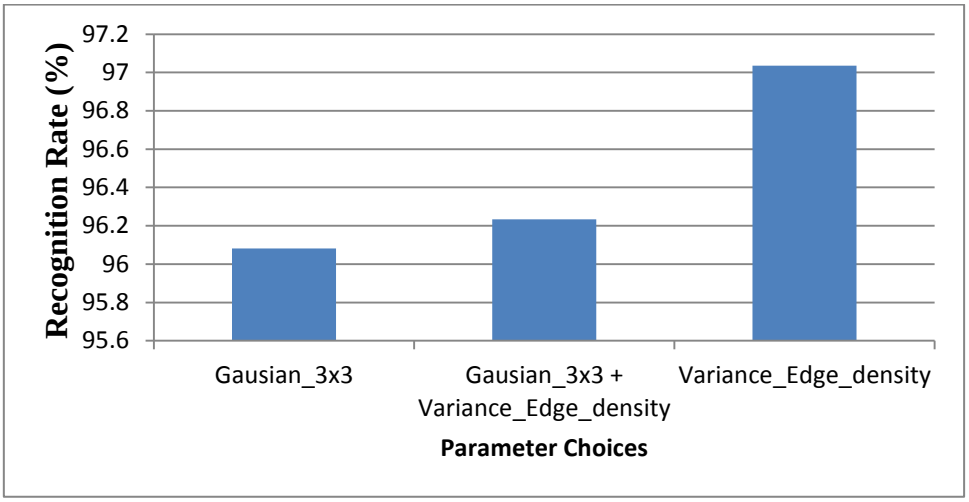


Figure 5.12. Comparison of the individual performance result of the 3x3 Gaussian and Variance with edge statistics parameter choices and the performance result of combination of the two.

In summary, the parameter choices when combined in pairs can each be employed effectively in an implementation of the proposed algorithm. In this research, the parameter choices believed to be effective in the implementation of the proposed structured HOG are:

- *Enhancement strategy* = none.
- *Number of bins* = 18.
- *Statistics* =Edge + Variance
- *Bin Voting* = Gradient magnitude

5.2.2 Algorithm performance evaluation Experiments

In the previous experiments, the basic aim was to select the best set of parameters that will produce the most effective result. In this section, experiments are conducted in order to compare the performance of the proposed structured HOG and the conventional HOG implemented with various cell divisions. For a fair comparison, both the proposed structured HOG algorithm and the conventional HOG are made to have the same values of all the common parameters. The common parameters and their values are:

- *Number of Bins* = 18
- *Normalisation*= l_2 normalisation
- *Bin voting* = Gradient magnitude

The performance criteria adopted is based on the following indicators:

- ❖ Computational complexity in terms of number of cells and feature vector length
- ❖ Detection/Recognition rates on the Database 2

5.2.2.1 Comparing computational complexity

To implement conventional HOG according to Dalal and Triggs [12], the computation of HOG solely depends on the size of the image and the cell size.

For example, considering an image of size 25 x 100, The HOG computation with 8 x 8 cell size and 50 % overlapping block will be: the **25 x 100** pixel image will be divided into **12.5** blocks across and **3.5** blocks vertically, for a total of **52** blocks. The **0.5** is for the half block that covers the cells at the edge. Each block contains **4** cells with **18**-bin histogram for each cell, for a total of **72** values per block. This brings the final vector size to **3.5** blocks across × **12.5** blocks vertically × **4** cells per block × **18**-bins per histogram = **3,150** values.

Considering a much larger cell size of **9** × **17**; this gives **6** cells across and **3** cells vertically, this cell division is closely similar to our proposed structured HOG in terms of cell division. The HOG computation will be = **5.5** blocks across × **2.5** blocks vertically × **4** cells per block × **18**-bins per histogram = **990** values.
As described in section (4.1.1.1) the computation of our proposed algorithm is simply = **324** values and the number of cells is fixed irrespective of the size of the image.

5.2.2.2 Experiment 5.16: Experiments to perform plate detection with conventional HOG (8 x 8 cell size)

The experiment was performed with the following parameters apart from default parameters adopted in all the previous experiments:

- *Block overlapping* = 2 x 2 block 50% overlap.

ANN training phase:

Table 5.46. Experiments conventional HOG (8 x 8 cell size) Gaussian filter Confusion matrix (in %) for Testing dataset (380 images)

	Positive	**Negative**	**Total**
Positive	100	0	
Negative	0.40	99.60	
Total			**99.74**

Table 5.47. Experiments conventional HOG (8 x 8 cell size) Combined confusion matrix (in %) for whole dataset (3796 images)

	Positive	**Negative**	**Total**
Positive	100	0	
Negative	0.847817	99.15	
Total			**99.47**

Scene car images testing phase:

Table 5.48. Experiments with conventional HOG (8 x 8 cell size) experimental results of running GA 5 times on Database 2

Result of 5 runs (%)					
1	**2**	**3**	**4**	**5**	**Average**
93.83	93.51	94.27	93.58	93.13	**93.66**

5.2.2.3 Experiment 5.17: Experiments to perform plate detection with conventional HOG (9 x 17 cell size)

The experiment was performed with the following parameters apart from default parameters adopted in all the previous experiments:

- *Block overlapping* = 2 x 2 block 50% overlap.

ANN training phase:

Table 5.49. Experiments conventional HOG (9 x 17 cell size) Gaussian filter Confusion matrix (in %) for Testing dataset (380 images)

	Positive	**Negative**	**Total**
Positive	99.36	0.64	
Negative	0	100	
Total			**99.74**

Table 5.50. Experiments conventional HOG (9 x 17 cell size) Combined confusion matrix (in %) for whole dataset (3796 images)

	Positive	**Negative**	**Total**
Positive	99.86	0.14	
Negative	0	100	
Total			**99.95**

Scene car images testing phase:

Table 5.51. Experiments with conventional HOG (9 x 17 cell size) experimental results of running GA 5 times on Database 2

Result of 5 runs (%)					
1	**2**	**3**	**4**	**5**	**Average**
94.08	94.53	94.21	93.45	94.02	**94.06**

5.2.2.4 Experiment 5.18: Experiments to perform plate detection with conventional HOG (25 x 10 cell size)

The experiment was performed with the following parameters apart from default parameters adopted in all the previous experiments:

- *Block overlapping* = 2 x 2 block 50% overlap.

ANN training phase:

Table 5.52. Experiments conventional HOG (25 x 10 cell size) Gaussian filter Confusion matrix (in %) for Testing dataset (380 images)

	Positive	Negative	Total
Positive	99.39	0.61	
Negative	0	100	
Total			**99.74**

Table 5.53. Experiments conventional HOG (25 x 10 cell size) Combined confusion matrix (in %) for whole dataset (3796 images)

	Positive	Negative	Total
Positive	99.79	0.21	
Negative	0.04	99.96	
Total			**99.89**

Scene car images testing phase:

Table 5.54. Experiments with conventional HOG (25 x 10 cell size) experimental results of running GA 5 times on Database 2

Result of 5 runs (%)					
1	**2**	**3**	**4**	**5**	**Average**
81.87	81.93	81.55	81.81	83.21	**82.07**

5.2.2.5 Discussions

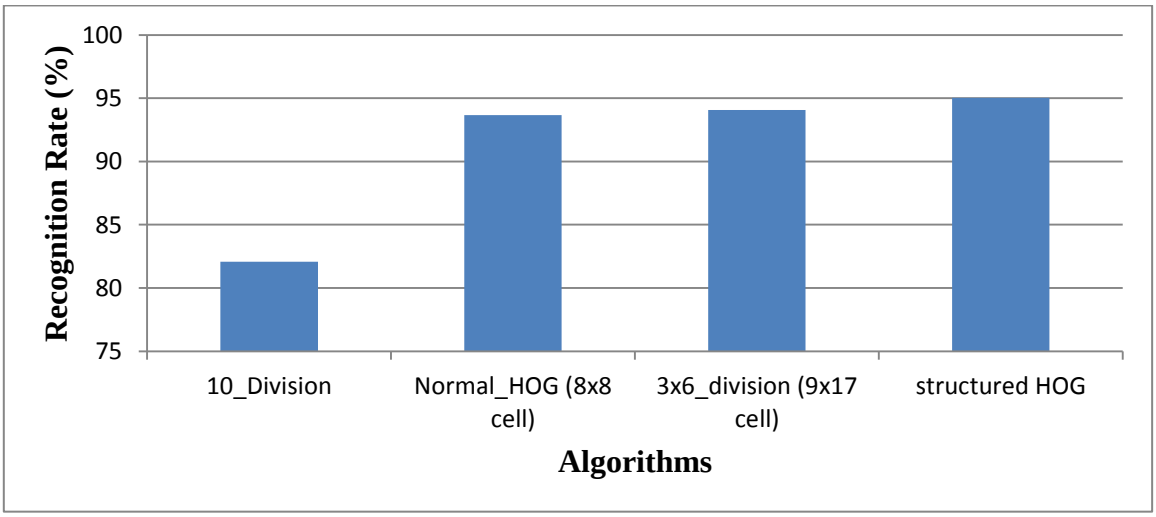


Figure 5.13 Comparison of the proposed structured HOG with conventional HOG at various cell divisions (cell sizes) with all implemented with exactly equal similar parameters

From the results obtained above, it can be concluded that the conventional HOG both from the classifier training result and GA detection results gives a low performance result compared to the structured HOG. Also, the 3x6 cell division HOG, which has a structure very similar to out proposed structured HOG but with higher computation and vector size than our structured HOG could also could not outperformed our proposed structured HOG.

5.2.3 Plate detection test running on non-Turkish plate database (Database 3)

In this section, the proposed plate detection algorithm is tested on a non-Turkish plate database. The experiment was performed with the classifier trained in *Experiment 5.10: Adding Variance and Edge density to Structured HOG)* of section (5.2.1.3.3).

The database used is **Database 3**: Real car scene images database (Non-Turkish)) of section (5.1.2.3). The table the below summarise the proposed algorithm results.

Table 5.55. Result of testing the proposed algorithm with Database 3(Non- Turkish plates, variable distance from the camera and at different environmental conditions).

Database Group	**Result of 5 runs;**							
	1	**2**	**3**	**4**	**5**	**Average**	**Total**	**Rate (%)**
Day Blurred	7	7	7	7	7	**7**	**7**	**100**
Day Colour	193	193	193	196	194	**193.8**	**205**	**94.55**
Day Grey Scale	44	45	45	45	45	**44.8**	**48**	**93.33**
Day Shadows in Plate	46	45	43	47	46	**45.4**	**49**	**92.65**
Day Very Close View	118	116	117	118	117	**117.2**	**122**	**96.07**
Difficult Dirt Shadows	144	145	142	141	143	**143**	**160**	**89.38**
Difficult Shadows	24	24	24	24	23	**23.8**	**26**	**91.54**
Difficult Tracks	24	19	23	21	25	**22.4**	**35**	**64**
Night Flash	3	3	3	3	3	**3**	**3**	**100**

It can be observed from the results obtained from this experiments that, most of the plate misdetection failures are mainly due to the variation of the size of the plate in the image as the GA was run with a fixed detecting window (fixed scale). However, it was noticed that as the GA is run on the images individually with an appropriate detecting window size, almost all the plate numbers can be detected from all the images in this database. Some of the difficult plates correctly detected by the proposed algorithm can be shown in Fig. 5.14.

Figure 5.14: Images of some successfully detected plate

5.3 Comparison of the proposed plate detection with some of published work in the literature

The previous section detailed the experiments carried out to determine the optimal set of parameters to be used for the implementation of our proposed algorithm. In this section, the results obtained by some of the published work in literature are compared with the best results obtained from our proposed algorithm. However, it is important to point out that there is no benchmark database that can be used for a fair comparison of the already published work with our proposed algorithm. The results presented are the published results by these authors.

Table 5.56. Comparison of some of already published work in the literature with our proposed algorithm

S/No.	Authors	Feature extraction Technique	No. of Images	Detection Rate
1.	Louka Dlagnekov (2004) [20]	Horizontal and vertical derivatives of plate image with Adaboost	158	95.6%
2.	Ho, W. T. et al. (2009) [21]	Adaboost+(SIFT+SVM)		90.185%
3.	Zheng, D. et al. (2005) [15]	Edge statistics	1134	99.8
4.	Martin, F., et al. (2002) [13]	Morphological Operation	105	87.61%
5.	Hongliang et al. (2004) [14]	Edge statistics + Morphological Operation	9825	99.6%
6.	Anagnostopoulos et al. (2006) [17]	sliding concentric windows	1334	96.5%
7.	Shyang-Lih et al. (2004) [18]	fuzzy disciplines attempts	1088	97.9%
8.	Azad, R. et al. (2013) [19]	HSV image transformation	150	99.3%
9.	Assis da Silva, et al. (2013) [22]	SIFT + template matching	420	99.52%
10.	**Our Proposed Method**	**Our Proposed Method**	**1572**	**97.03%**

CHAPTER SIX
CONCLUSION AND FUTURE WORK

6 Conclusion

The basic aim of this thesis research was to develop an algorithm that will accurately detect the location of a license plate in an image. During the research study, as presented in the previous chapters, an algorithm was developed that can specifically extract the features of a plate by adopting the structure of the plate. With this, we can gladly states that the aim had been excellently accomplished with a successful plate detection of **1526** out of **1572** car images (**97.03 %**). The only **46** plates not detected where attributed to mainly due to the presence of a highly similar textual objects present in the car images and a rare instance of size of the plates in the images. Furthermore, the performance of the proposed algorithm was tested against highly dynamic car images of plates not even similar to the plates images used in training the classifier with images captured at various environmental conditions, different camera distance and poor quality plate images. Still, the overall detection performance was impressive with plate detection of almost **601** out of **655** images (almost **92%**). The only **54** plates not detected where mainly due to size of the plates in the images because of the varying camera distances. The performance of the algorithm if compared with some already published work in the literature can be observed, as shown in Table 5.56.

During the research evaluation experiments, in determining set of best parameters suitable for the proposed algorithm, correlation problem was observed between parameters. The parameter selection experiments were conducted with some parameters fixed and the parameter of interest varies. However, as the selected best parameters where combined, the performance was reduced. The possible reason observed for the reduction had been explained in detail in the previous chapter.

We can conclude in general that the research is successful even though it was made to be developed in real-time implementation but was finally developed and implemented on MATLAB software environment due to time constraint.

6.1 Future Work

Although, the aim of the research had been accomplished, however, what is generally not considered in the research is the detection running time for a successful detection of a plate; this is because the proposed algorithm was implemented with MATLAB programming codes and not with real time codes such as C++ that the running time will be fairly studied. Future work could be a full implementation of the proposed algorithm on a real-time programming paradigm such as C++ and a closely study of the running detection time.

Another important aspect not considered in this research is the method of implementing the proposed structured HOG. As with the conventional HOG, when implemented and used on a portion of a large image, the histograms is repeatedly calculated on already visited portion of the image, this is because the detection window is moved across image from one pixel to another either sequentially with repeated pixel overlapping or randomly with sampled points. To tackle this repeated computation to minimal repetitions, future work could employ the concept of Integral histogram [49].

Also as stated in the previous chapters, this research focuses on only one of part ALPR system; future research could be integration of a complete ALPR system involving all the components of the system.

REFERENCES

1. Kim, S. K., Kim, D. W., & Kim, H. J. (1996, September). A recognition of vehicle license plate using a genetic algorithm based segmentation. In *Image Processing, 1996. Proceedings., International Conference on* (Vol. 1, pp. 661-664). IEEE.
2. H S. Kim "Recognition of a car number plate by a neural network," Proc. of Korea Information Science Society(K1SS) fall conference, vol. 18, no. 2, pp. 259-262, 1991
3. B T Cheon "The extraction of a number plate from a moving car," Proc of First Workshop on Character Recognition, pp 133-136, 1993.
4. H S Chong and J Cho, "Locating Car License Plate using Sub region Feature," Journal of the KISS, vol 21, no 6, pp 1149-1159, 1994.
5. The Wired Repo Man: He's Not 'As Seen on TV'". *New York Times*. 28 February 2010. Retrieved 2012-01-24.
6. UK Billboards Equipped with License Plate Spy Cameras". TheNewspaper.com. 25 September 2009. Retrieved 2013-07-02.
7. Friedrich, M., Jehlicka, P., & Schlaich, J. (2008, May). Automatic number plate recognition for the observance of travel behavior. In *Processding of 8th International Conference on Survey Methods in Transport* (pp. 1-17).
8. "CCTV network tracks 'getaway' car". BBC News. 2005. Retrieved 2013-08-12.
9. Keilthy, L. (2008). ANPR system performance. *Parking trend international*. http://www.docstoc.com/docs/21737678/Measuring-ANPR-System-Performance.
10. Sexton, Steve (3 July 2003). "License-plate spray foils traffic cameras". *Washington Times*. Retrieved 2013-07-02.
11. Anagnostopoulos, C. N., Anagnostopoulos, I. E., Psoroulas, I. D., Loumos, V., & Kayafas, E. (2008). License plate recognition from still images and video sequences: A survey. *Intelligent Transportation Systems, IEEE Transactions on*, *9*(3), 377-391.
12. Dalal, N., & Triggs, B. (2005, June). Histograms of oriented gradients for human detection. In Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on, 1(1), 886-893. IEEE.
13. Martin, F., Garcia, M., & Alba, J. L. (2002, June). New methods for automatic reading of VLP's (Vehicle License Plates). In Proc. IASTED Int. Conf. SPPRA (pp. 126-131).
14. Hongliang, B., & Changping, L. (2004, August). A hybrid license plate extraction method based on edge statistics and morphology. In Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on, 2(2), 831-834. IEEE.

15. Zheng, D., Zhao, Y., & Wang, J. (2005). An efficient method of license plate location. Pattern Recognition Letters, 26(15), 2431-2438.
16. Wang, S. Z., & Lee, H. J. (2003, October). Detection and recognition of license plate characters with different appearances. In Intelligent Transportation Systems, 2003. Proceedings. 2003 IEEE 2(2), 979-984. IEEE.
17. Anagnostopoulos, C. N. E., Anagnostopoulos, I. E., Loumos, V., & Kayafas, E. (2006). A license plate-recognition algorithm for intelligent transportation system applications. Intelligent Transportation Systems, IEEE Transactions on, 7(3), 377-392.
18. Chang, S. L., Chen, L. S., Chung, Y. C., & Chen, S. W. (2004). Automatic license plate recognition. Intelligent Transportation Systems, IEEE Transactions on, 5(1), 42-53.
19. Azad, R., Davami, F., & Azad, B. (2013). A novel and robust method for automatic license plate recognition system based on pattern recognition. Advances in Computer Science: an International Journal, 2(3), 64-70.
20. Dlagnekov, L. (2004). License plate detection using adaboost. Computer Science and Engineering Department, San Diego.
21. Ho, W. T., Lim, H. W., & Tay, Y. H. (2009, April). Two-stage license plate detection using gentle Adaboost and SIFT-SVM. In Intelligent Information and Database Systems, 2009. ACIIDS 2009. First Asian Conference on (pp. 109-114). IEEE.
22. Assis da Silva, F., Olivette Artero, A., Stela Veludo de Paiva, M., & Barbosa, R. L. (2013). ALPRS-A New Approach for License Plate Recognition using the Sift Algorithm.
23. Gonzalez, R. C., Woods, R. E., & Eddins, S. L. (2004). Digital image processing using MATLAB. Pearson Education India.
24. Nixon, M., & Aguado, A. S. (2008). Feature extraction & image processing. Academic Press.
25. Dunteman, G. H. (1989). Principal components analysis (No. 69). Sage.
26. Loog, M., & de Ridder, D. (2006, August). Local discriminant analysis. In Pattern Recognition, 2006. ICPR 2006. 18th International Conference on, 3(1), 328-331. IEEE.
27. Ahonen, T., Hadid, A., & Pietikäinen, M. (2004). Face recognition with local binary patterns. In Computer vision-eccv 2004 (pp. 469-481). Springer Berlin Heidelberg.
28. Bay, H., Tuytelaars, T., & Van Gool, L. (2006). Surf: Speeded up robust features. In Computer Vision–ECCV 2006 (pp. 404-417). Springer Berlin Heidelberg.

29. Lowe, D. G. (1999). Object recognition from local scale-invariant features. In Computer vision, 1999. The proceedings of the seventh IEEE international conference on 2(1), 1150-1157. IEEE.
30. Mitchell, T. M. (1997). Machine learning. 1997. Burr Ridge, IL: McGraw Hill, 45.
31. Cortes, C., & Vapnik, V. (1995). Support-vector networks. Machine learning, 20(3), 273-297.
32. Yegnanarayana, B. (2009). Artificial neural networks. PHI Learning Pvt. Ltd.
33. Friedman, N., Geiger, D., & Goldszmidt, M. (1997). Bayesian network classifiers. Machine learning, 29(2-3), 131-163.
34. Klir, G. J., & Yuan, B. (1995). Fuzzy sets and fuzzy logic 4(4). New Jersey: Prentice Hall.
35. Freund, Y., Schapire, R., & Abe, N. (1999). A short introduction to boosting. Journal-Japanese Society For Artificial Intelligence, 14(771-780), 1612.
36. Russell, S. J., Norvig, P., Canny, J. F., Malik, J. M., & Edwards, D. D. (1995). Artificial intelligence: a modern approach (Vol. 2). Englewood Cliffs: Prentice hall.
37. KaewTraKulPong, P., & Bowden, R. (2002). An improved adaptive background mixture model for real-time tracking with shadow detection. In Video-Based Surveillance Systems (pp. 135-144). Springer US.
38. Zivkovic, Z. (2004, August). Improved adaptive Gaussian mixture model for background subtraction. In Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on (Vol. 2, pp. 28-31). IEEE.
39. Stauffer, C., & Grimson, W. E. L. (1999). Adaptive background mixture models for real-time tracking. In Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on. (Vol. 2). IEEE.
40. Lee, E. R., Kim, P. K., & Kim, H. J. (1994, November). Automatic recognition of a car license plate using color image processing. In Image Processing, 1994. Proceedings. ICIP-94., IEEE International Conference (Vol. 2, pp. 301-305). IEEE.
41. Wang, T. H., Ni, F. C., Li, K. T., & Chen, Y. P. (2004, March). Robust license plate recognition based on dynamic projection warping. In Proc. IEEE Int. Conf. Netw., Sens. Control (pp. 784-788).
42. Duan, T. D., Du, T. H., Phuoc, T. V., & Hoang, N. V. (2005, February). Building an automatic vehicle license plate recognition system. In Proc. Int. Conf. Comput. Sci. RIVF (pp. 59-63).

43. Nijhuis, J. A. G., Ter Brugge, M. H., Helmholt, K. A., Pluim, J. P. W., Spaanenburg, L., Venema, R. S., & Westenberg, M. A. (1995, November). Car license plate recognition with neural networks and fuzzy logic. In Neural Networks, 1995. Proceedings., IEEE International Conference on (Vol. 5, pp. 2232-2236). IEEE.
44. Ter Brugge, M. H., Stevens, J. H., Nijhuis, J. A. G., & Spaanenburg, L. (1998, April). License plate recognition using DTCNNs. In Cellular Neural Networks and Their Applications Proceedings, 1998 Fifth IEEE International Workshop on (pp. 212-217). IEEE.
45. Taleb-Ahmed, A., Hamad, D., & Tilmant, G. (2003, June). Vehicle license plate recognition in marketing application. In Intelligent Vehicles Symposium, 2003. Proceedings. IEEE (pp. 90-94). IEEE
46. Wu, C., On, L. C., Weng, C. H., Kuan, T. S., & Ng, K. (2005, August). A Macao license plate recognition system. In Machine Learning and Cybernetics, 2005. Proceedings of 2005 International Conference on (Vol. 7, pp. 4506-4510). IEEE.
47. Møller, M. F. (1993). A scaled conjugate gradient algorithm for fast supervised learning. Neural networks, 6(4), 525-533.
48. Media Lab LPR Database, http://www.medialab.ntua.gr/research/LPRdatabase.html Retrieved 2014-05-14.
49. Porikli, F. (2005, June). Integral histogram: A fast way to extract histograms in cartesian spaces. In Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on, 1(1), 829-836. IEEE.
50. Whitley, D. (1994). A genetic algorithm tutorial. Statistics and computing, 4(2), 65-85
51. Hung, K. M., & Hsieh, C. T. (2010). A real-time mobile vehicle license plate detection and recognition. Tamkang Journal of Science and Engineering, 13(4), 433-442.
52. Zheng, K., Zhao, Y., Gu, J., & Hu, Q. (2012, May). License plate detection using haar-like features and histogram of oriented gradients. In Industrial Electronics (ISIE), 2012 IEEE International Symposium on (pp. 1502-1505). IEEE.
53. Mousa, A. (2012). Canny Edge-Detection Based Vehicle Plate Recognition. International Journal of Signal Processing, Image Processing & Pattern Recognition, 5(3).
54. Ji-yin, Z., Rui-rui, Z., Min, L., & Yin, L. (2008, December). License plate recognition based on genetic algorithm. In Proceedings of the 2008 International Conference on Computer Science and Software Engineering on, 1(1), 965-968. IEEE Computer Society.

55. Soon, C. K., Lin, K. C., Jeng, C. Y., & Suandi, S. A. (2012). Malaysian Car Number Plate Detection and Recognition System. Australian Journal of Basic & Applied Sciences, 6(3).