بسم الله الرحمن الرحيم

Palestine Polytechnic University

College of Engineering and Technology
Electrical and Computer Engineering Department

Graduation Project

Palestinian License Plate Recognition System (PLPRS)

Project Team

Amal S.Qeseya **Rana R. Sneineh**

Sahar S.Salhab

Project Supervisor
Dr.Hashem Tamimi
Dr. Ala'a Al_Halawani

Hebron – Palestine
2008

Palestinian License Plate Recognition System (PLPRS)

Project Team:

Amal S.Qeseya

Rana R.Sneineh

Sahar S.Salhab

Supervisor:

Dr.Hashem Tamimi

Dr. Alaa H.Halawani

Graduation Project Report

Submitted to the Department of Electrical and Computer Engineering in the College of Engineering and Technology

Palestine Polytechnic University

College of Engineering and Technology

Electrical and Computer Engineering Department

Hebron – Palestine

2008

جامعة بوليتكنك فلسطين
الخليل-فلسطين

كلية الهندسة و التكنولوجيا

دائرة الهندسة الكهربائية والحاسوب

اسم المشروع

Palestinian License Plate Recognition System (PLPRS)

أسماء الطلبة

أمل صبحي قيسية رنا ربحي ابو سنينه سحر سميح سلهب

بناءً على نظام كلية الهندسة والتكنولوجيا وإشراف ومتابعة المشرف المباشر على المشروع وموافقة أعضاء اللجنة الممتحنة تم تقديم هذا المشروع إلى دائرة الهندسة الكهربائية والحاسوب، وذلك للوفاء بمتطلبات درجة البكالوريوس في الهندسة تخصص هندسة أنظمة الحاسوب.

توقيع المشرف

....

توقيع اللجنة الممتحنة

....

توقيع رئيس الدائرة

....

Abstract

This project's aim is to design a vehicle access control system for a Palestinian organization. Such a system is built on automatic identification and recognition of vehicles based on their plate numbers so as to restrict the entrance for only authorized vehicles. The recognition of the plates numbers are accomplished through four major steps which are: Image acquisition, next license plate extraction, then plate segmentation, and finally, the character recognition step.

Specific algorithms and techniques for identifying the vehicles have been proposed and implemented. As for the extraction phase the vertical edge matching is applied. The segmentation stage is performed using the horizontal and vertical projection profile. And finally, the neural approach is applied for the recognition phase.

The proposed system has been implemented using visual C ++ 2005, and OPENCV library, simulation was done using Matlab 7.0.1. The performance of the system has been investigated on images of about 400 vehicles captured under various conditions. Results have shown the capability of this system in identifying 96% of the images correctly proving the system to be quite efficient.

الملخص

يهدف هذا المشروع إلى تصميم نظام أمني لمؤسسة فلسطينية للسماح بدخول السيارات المخول لها فقط بالدخول من خلال قراءة أرقام لوحات السيارات و الذي يعتمد على تصميم نظام لقراءة لوحات ترخيص السيارات الفلسطينية. يتكون أي نظام حاسوبي لقراءة لوحات ترخيص السيارات من أربعة مراحل أساسية. أول هذه المراحل هي التقاط الصورة، المرحلة الثانية هي استخلاص اللوحة، تهدف هذه المرحلة إلى تحديد موقع اللوحة في الصورة المأخوذة. المرحلة الثالثة تعمل على تجزئة اللوحة إلى أرقام منفردة.ثم قراءة اللوحة للتعرف عليها و هي المرحلة النهائية في تصميم هذا النظام.

تم تطبيق طرق حاسوبية مختلفة لتحقيق هذه المراحل، ففي مرحلة استخلاص اللوحة تم تطبيق مطابقة الحواف الرأسية للوحة. في مرحلة تجزئة اللوحة تم تطبيق صور الإسقاط الرأسية و الأفقية. أخيرا لإتمام المرحلة النهائية تم استخدام الشبكات العصبية للقيام بعملية التعرف النهائي و قراءة لوحة الترخيص.

هذا و فد تم بناء تطبيق حاسوبي كنظام لقراءة لوحات الترخيص الفلسطينية باستخدام برنامج فيجوال سي ++ نسخة 2005, وتم اختيار ماتلاب النسخة 7.0.1 لاختبار هذا النظام قبل تطبيقه. تم اختبار هذا النظام على صور 400 سيارة تم التقاطها في ظروف مختلفة, و قد أظهرت النتائج قدرة النظام على قراءة لوحة الترخيص بصورة صحيحة في 96% من هذه الصور.

Dedication

They Were Always There When Dark Has Fallen to

Burn a Candle.

To our beloved Parents.

Acknowledgment

In the name of Allah, Most Gracious, Most Merciful

All praise and glory to Almighty Allah who gave us courage and patience to carry out this work. Peace and blessing of Allah be upon last Prophet Muhammad (Peace Be upon Him).

To our supervisors, Dr. Alla Halawani & Dr.Hashem Tamimi for their insightful inputs, valuable discussions, & the assistance they have provided throughout the lifetime of this project.

To Mr.Majdi Zalloum for his continous help and generous support.

To Eng.Sami salameen & Eng.Ahmad Tamimi for the efforts they have paid & the valuable contribution they have generously offered to accomplish this work.

To Abeer, Suhaila, Suha, Nada, Sahar, Heba, & Maison for their continuous encouragement, and priceless support.

Very special thanks to our families for their priceless patience and for offering words of encouragements to spur our spirits at moments of depression.

To our friends and colleagues with whom we spent the best times of our lives.

Thank you

The Team:

Amal Qeseya

Rana Sneineh

Sahar Salhab

Contents

Chapter One: Introduction

Chapter Tow: Theoretical Background

Chapter Three: Conceptual Design

Chapter Four: Implementation

Chapter Five: Experimental analysis and result

Chapter six: Conclusions and future work

List of Tables

Table **Page**

List of Figures

Figure **Page**

List of Lists

List of Lists **Page**

CHAPTER ONE 1

Introduction

1.1 Introduction.

1.2 General idea about the project.

1.3 Palestinian License Plate description.

1.4 Project objectives.

1.5 Literature review.

1.6 Application of license plate recognition system.

1.7 Cost estimation.

1.8 Time Chart.

1.9 Report organization

CHAPTER ONE
Introduction

1.1 Introduction

With the rapid development of intelligent security systems, automatic identification of vehicles using license plates has played a significant role in a lot of applications. The License Plate Recognition system is a system in which image processing technology is applied in order to extract the features, segment, recognize characters and identify those vehicles on the basis of pattern recognition techniques. Much work has been done for the recognition of European, Korean, Chinese and other license plates. However, little work has been done for the Palestinian plates.

1.2 General idea about the project

The project's main idea is designing an automatic Palestinian license plate recognition system that has the capability of identifying vehicles according to their plates numbers, that could be implemented in different applications where security is a concern. The selected application that was chosen is concerned in controlling and allowing the access of the organization vehicles.

Automatic license plate recognition system has the capability of recognizing the license plate number using the image-processing techniques to obtain the quantitative information that is used to control the process of opening and closing the gates of governmental departments, universities, etc. The proposed system's design contains a photocell that detects the arrival of the vehicles and when it does it sends a signal to the connected computer which in return initiates a trigger command to a web camera to capture the image of the object to make the necessary analysis to locate and recognize the license plate number. Then the system compares the number it has with several numbers stored in a database. If the number matches any of the stored numbers it sends a signal to the DC motor to open the gate, otherwise the gate is not opened.

The structure of the proposed system depend heavily on image processing analysis techniques to recognize the license plates. Input to this system is an image captured by a digital camera that contain license plate, and the output is the recognition of characters on the license plate. To achieve this, image analysis must pass through four steps which are: image preprocessing, license plate extraction, segmentation, and character recognition. The sequence of these steps is shown in Figure 1.1. The first task is pre-processing the captured image. The second The second task extracts the region that contains license plate. The third task segments the plate into seven blocks of individual characters as the Palestinian plate contains seven characters. Finally, the last task identifies the segmented characters.

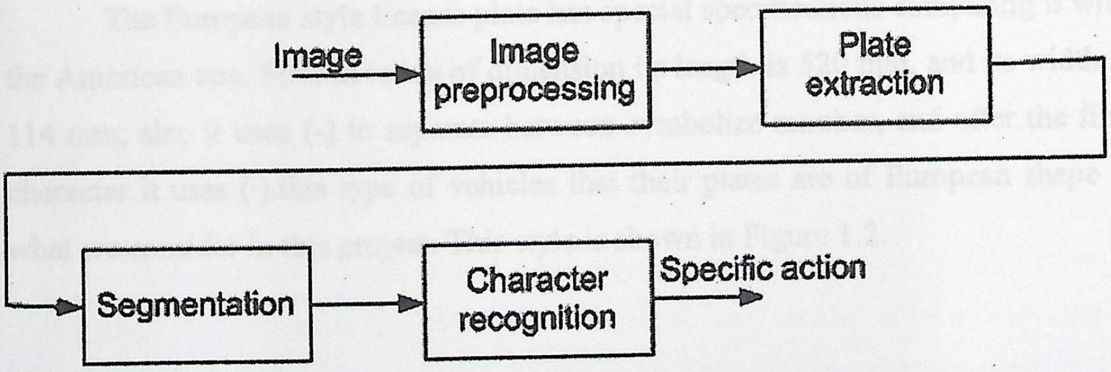


Figure 1.1 Structure of the proposed system

This chapter will initially discuss the available shapes of Palestinian license plate, and then it focuses on the objectives of license plate recognition system. Next, it reveals previous work that has been done in this area, following that it lists a number of applications in which this system can be used. Finally, cost estimation, and report organization are addressed.

1.3 Palestinian License Plate description

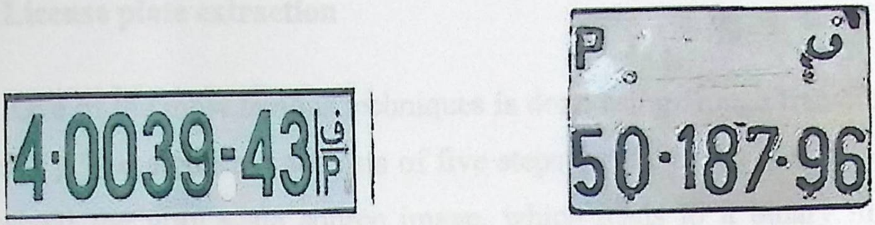
As known that there are two styles adopted for the Palestinian license vehicle plates: the European style license plate, and the American style license plate. Each has its own descriptions that distinguish it, but in general they share the same properties regarding the numbers and the symbols they contain. Each one of them includes seven characters (numbers from 0-9), and both have the same distinguishing symbols (P,ف) indicating that those vehicles are Palestinian plates, these symbols appear in the left or at the top of the plates. The rest of this section discusses the differences between the two shapes.

1.3.1 European style license plate

The European style license plate has special specifications comparing it with the American one. First in terms of dimension its length is 520 mm, and its width is 114 mm, also it uses (-) to separate between symbolize number, and after the first character it uses (·).this type of vehicles that their plates are of European shape is what we consider in this project. This style is shown in Figure 1.2.

1.3.2 American style license plate

The American style license plates also have their own distinctive features. Regarding the dimensions the length is 340 mm, and its width is 220 mm, uses (·) to separate between symbolize numbers. As shown in Figure 1.2.



(a):European style license plate

(b): American style license plate

Figure 1.2 Palestinians license plate adopted styles

1.4 Project objective

This project clearly aims to achieve the following goals.

- Designing a system that has the capability of reading and identifying the Palestinian license plate is the project's main objective.
- Build a vehicle access-control system for an organization based on the automatic recognition of the plates. This system at first read and identify the plates numbers of the vehicles, then it determines whether these vehicles are allowed to access or not.
- Build a reliable security system that delivers the optimal performance in terms of accuracy.

1.5 Literature Review

The accuracy of the license plate recognition system depends heavily on extraction, segmentation, and character recognition. This section gives an overview of the research carried out so far to achieve these steps and the techniques that have been used to identify the license plate.

1.5.1 License plate extraction

One of the most famous techniques is done using Hough transforms as done by Kim [1]. The algorithm consists of five steps for the extraction. The first step is to threshold the gray scale source image, which leads to a binary image. In the second step the resulting image is passed through two parallel sequences in order to extract the horizontal and vertical line segments respectively. The result is an image with edge highlighted. In the third step the resultant image is used as an input to the Hough transforms, this produces a list of lines in the form of accumulator cells. In fourth step, the above cells are analyzed and line segments are computed. Finally the list of horizontal and vertical line segment is combined and any rectangular regions matching the dimensions of the plate are kept as candidate regions. The disadvantage is that, this method has high memory requirements.

Park et.al, Kim et.al, & Jung et.al [2], showed that the extraction of the plates could be done using two neural networks-based filters and a post processor to combine two filtered images in order to locate the plate. The two neural networks used are vertical and horizontal filters, which examine small windows of vertical and horizontal cross sections of an image and decide whether each image contains the plate. Cross sections have sufficient information for distinguishing the plate.

Duan et.al, Hong et.al, Phuoc et.al, & Nguyen et.al [3] did their project based on the combination of Hough transform and contour algorithm. First, they used the contour algorithm to detect closed boundaries of objects. These contour lines are transformed to Hough coordinates to find two interacted parallel lines (one of 2 parallel lines holds back the other 2-parallel lines and establishes a parallelogram form object) that are considered as a plate-candidate. After having those candidates the ratios between widths and heights are applied to select the appropriate plate.

1.5.2 Segmentation

This section discusses previous work done for the segmentation of characters. Different approaches have been proposed such as; Gonzalez et.al, & Woods et.al [4] suggested region growing for segmentation of characters. The idea behind this method is to identify one or more criteria that are characteristic for the desired region. After defining the criteria, the image is searched for any pixels that satisfy the requirements and whenever a pixel is encountered, its neighbors are examined, and if any also match the criteria both the pixels are considered to have the same region. Morel et.al [5] used partial differential equations- based technique. Hansen et.al [6] performed a horizontal projection of a thresholded plate. The approach is to search for changes from valleys to peaks, simply by counting the number of black pixels per column in the projection. A change from a valley to a peak indicates the beginning of a character, and vice versa.

1.5.3 Recognition

This section present previous approaches that were used to classify then recognize the individual characters. The classification is based upon extracted features, and then these features are classified using either syntactical or neural approaches.

Some of the previous work regarding recognition process is as follows: Mei Yu et.al [7] and Natio et.al [8] applied syntactical approach they used template matching. Matching includes the use of a database of characters or templates. This is a separate template for each possible input character, recognition is done by comparing the current input to each template in order to find the match. Hu [9] proposed seven moment that can be used as features to classify characters. These moments are invariant to scaling, rotation and translation. The obtained moments act as the features, which are then passed to the neural network for classification or recognition of characters.

1.6 Applications of License Plate recognition system

License plate recognition technology has been used around the world for a variety of applications. These applications can fall into different categories each to achieve a special purpose as a solution of important problems. The rest of this section mentions some of these applications.

- **Law Enforcement**

  The plate recognition system is used in order to produce a violation on speeding vehicles, illegal use of bus lanes. These speeding cars and illegal movements are usually caught by the traffic camera. Theses images are then processed in order to get the plate number to identify the violators.

- **Border Crossing**

  This application helps the registry of entry or exits to a country, and can be used to monitor the border crossings. Each vehicle information is registered into a central database with additional information if necessary.

- **Parking**

  The license plate recognition system is used to automatically enter pre-paid members and calculate parking fee for non-members. The car plate is recognized and stored and upon its exit the car plate is read again and the driver is charged for the duration of parking for those non-members.

- **Security and Access Control**

This is applicable in areas such as governmental institutions, military areas where a high level of security is required. License plate recognition system would be considered as a subsystem of the overall system. And its used to let only the authorized members to enter by checking their plate number with additional information, these information are then processed and fed into a database as input to find a match, to allow the entrance.

1.7 Cost Estimation

The following table shows the estimated total cost for the different components in this project.

EQUIPMENT	Estimated Cost($)
Web Camera	20
Photocell	18
DC motor	29
computer Software & Services	100
Different Tools	150
Total Cost	317

Table1.1 Project equipments estimated cost

1.8 Time Chart

The following Gannt chart shows the activities, efforts, and estimated time for each one.

Task \ Week	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Planning	▨	▨	▨	▨												
Analysis			▨	▨	▨	▨										
Design						▨	▨	▨	▨	▨	▨	▨				
Implementation										▨	▨	▨	▨	▨		
Testing										▨	▨	▨	▨	▨	▨	
Documentation & Presentation			▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	

Table1.2 Time chart

1.9 Report organization

This project presents a complete system for the recognition of the Palestinians license plate; the project is organized as follows:

Chapter one provides a general idea about the proposed system, a brief literature review of the techniques used in the phases of the LPR system, it also mentions some applications at where this system can be used.

Chapter two describes project theories, methods and techniques presented in this chapter, a brief introduction to nearest neighbor, and neural network are given also. The required hardware and software to build the whole system are describes in the rest of this chapter.

Chapter three handles in details the actual design for the proposed LPR system. It discusses the structure, and the techniques used for each of the system phases in details.

Chapter four shows hardware connection and pseudocode of the system phases are written in this chapter.

Chapter five discusses the experimental analysis and results obtained at different stages under different conditions.

Chapter six addresses the conclusions about the work, recommendations for developing this system.

CHAPTER TWO 2

Theoretical Background

2.1 Overview.

2.2 Image processing

2.3 Neural networks.

2.4 Hardware Tools.

2.5 Software Tools.

2.6 Summary

CHAPTER TWO
Theoretical Background

2.1 Overview

This chapter briefly talks about image processing, and then it introduces the theoretical background that would be used in this project as, thresholding, sobel operator, morphological operation, and neural networks. Finally, it lists the software and hardware tools that will be used in the design.

2.2 Image processing

Image processing is any form of information processing for which the input is an image such as photographs or frames and the output can either be an image or set of features related to the image. Most image processing technique involves treating the image as a two dimensional array of pixels, each pixel represents a sample of the original image, the intensity of each pixel is variable; in color images each pixel has three components such as red, green, and blue (the RGB model).

2.2.1 Thresholding

Thresholding is the simplest method used in image segmentation. It essentially involves changing the color or grey images into a binary image (binary image is a digital image stored as a logical array of '0' or'1', each pixel value can be one of the two values). Thresholding is done by allocating every pixel in the image either black or white depending on their value. The value that is used to determine whether any pixel is black or white is called threshold.

The process is as follows if the pixel's value is greater than the threshold value than set it to 1 (white), otherwise set it to 0 (black) the result is a black and white image, black pixels correspond to background and white pixels correspond to foreground or vice versa. However the most important step in this process is the selection of the threshold value. Several different methods for choosing the threshold value exist. One method of achieving this is done by choosing the mean value or median value, the justification behind this method is: if the pixels are brighter than the background they should also be brighter than the average [10]. This can work well in a noiseless image with a uniform background. However this will not be the case. Figure 2.1 show two images, before thresholding, and the other after thresholding.

If the original image is F, the output (thresholded) image B is calculated as shown in the following equation 2.1.

$$\forall i, j \,, \; i \in [1-M], j \in [1-N]$$

$$B[i\,,j] = \begin{cases} 1 \text{ if } F[i\,,j] \geq T \\ 0 \text{ if } F[i\,,j] < T \end{cases} \qquad \textbf{(2.1)}$$

Where M and N are image dimensions, and T is the threshold value. 1 stands for white and 0 for black.

either black or white depending on their value. The value that is used to determine whether any pixel is black or white is called threshold.

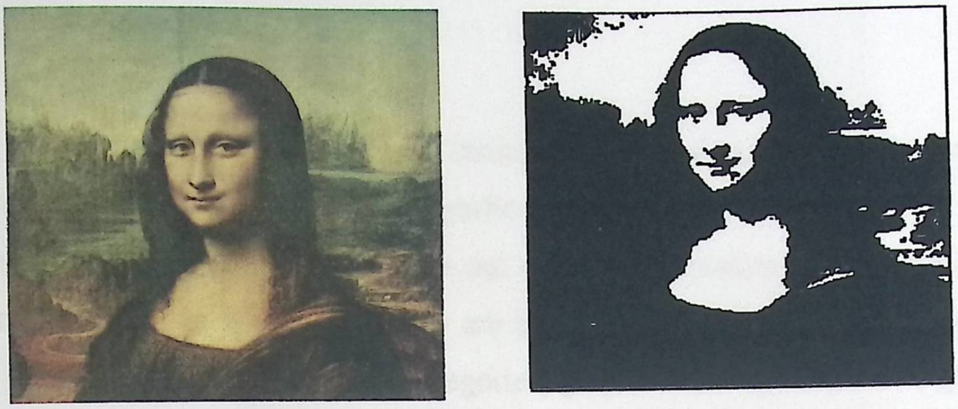
The process is as follows if the pixel's value is greater than the threshold value than set it to 1 (white), otherwise set it to 0 (black) the result is a black and white image, black pixels correspond to background and white pixels correspond to foreground or vice versa. However the most important step in this process is the selection of the threshold value. Several different methods for choosing the threshold value exist. One method of achieving this is done by choosing the mean value or median value, the justification behind this method is: if the pixels are brighter than the background they should also be brighter than the average [10]. This can work well in a noiseless image with a uniform background. However this will not be the case. Figure 2.1 show two images, before thresholding, and the other after thresholding.

If the original image is F, the output (thresholded) image B is calculated as shown in the following equation 2.1.

$$\forall i,j \, , \; i \in [1-M], j \in [1-N]$$

$$B[i\,,j] = \begin{cases} 1 \text{ if } F[i\,,j] \geq T \\ 0 \text{ if } F[i\,,j] < T \end{cases} \qquad \textbf{(2.1)}$$

Where M and N are image dimensions, and T is the threshold value. 1 stands for white and 0 for black.



(a): Original Image **(b): Thresholded Image**

Figure 2.1 Image thresholding

2.2.2 Convolution

Convolution is a simple mathematical operation which is fundamental to many common image processing operators. Convolution is done by multiplying each pixel in the input image by a numerical matrix of some finite size and shape (commonly referred to as the convolution kernel), This kernel slides over an area of the input image, generally starting at the top left, changes that pixel's value and then shifts one pixel to the right and continues to the right until it reaches the end of a row. It then starts at the beginning of the next row and so on until all pixels of the input image are manipulated.

If the kernel is given by h[j,k], {j=0,1,...,J-1; k=0,1,...,K-1}, and the input image is given by a[m,n], where m, and n are image dimensions, the convolution can be written as the following finite sum:

$$c[m,n]= a[m,n]\otimes h[m,n]= \sum_{j-0}^{j-1}\sum_{k=0}^{k-1} h[j,k]a[m-j,n-k] \qquad (2.2)$$

2.2.3 Edge detection

Edges in images are areas with strong intensity contrasts that is the intensity from one pixel to the next changes significantly [11]. Edge detecting in an image reduces the amount of data and filters out useless information, while keeping the important features in an image. There are many ways to perform edge detection. These methods lie in two different categories, gradient and laplacian. The gradient method detects the edges by looking for the maximum and minimum in the first derivative of the image. The Laplacian method searches for zero crossings in the second derivative of the image to find edges. Here in this project we will consider gradient methods and to be more specific we will choose sobel operator.

2.2.4 Gradient Methods

As mentioned earlier the gradient method detects edges by looking for the maximum and minimum in the first derivative of the image. Gradient methods detect the edges by performing a pair of 3×3 convolution masks [11]. Usually gradient detectors uses two convolution kernels to detect in vertical and another for horizontal, these kernels are called Gx and Gy respectively. These kernels can be applied separately at input image, to produce separate measurements of the gradient component. The combination of these two kernels will find the absolute magnitude at each point and the orientation of the gradient.

The gradient magnitude is given by equation 2.3, and its orientation is given by equation 2.5.

Gradient methods can be performed using different edge detector operators such as, Sŏbel, Prewitt, canný operators. Sobel and Prewitt are simple, fast, both are used to detect fine edges, and when the noise level is low. While Canny operator can be used to detect fine and rough edges at different noise levels. In this project

Sobel operator will be used, because its efficient, simple, and proved to give good results.

$$|G|= \sqrt{(Gx^2 + Gy^2)}$$ **(2.3)**

An approximate magnitude is computed using equation2.3:

$$|G|=|Gx| + |Gy|$$ **(2.4)**

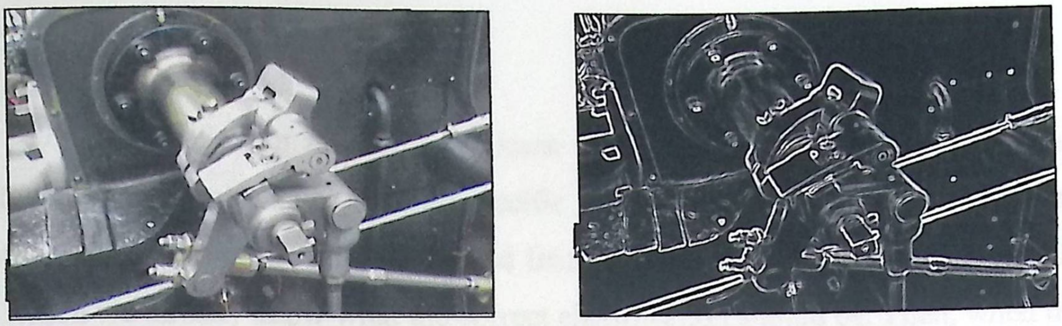
This is much faster to compute.

The angle of orientation of the edge is given in equation 2.4

$$\theta=\arctan(Gy/Gx)$$ **(2.5)**

2.2.4.1 Sobel Operator

The sobel operator performs a 2-D spatial gradient measurement on an image. It is used to find the approximate absolute gradient magnitude at each point in an input grayscale image. The Sobel edge detector uses a pair of 3x3 convolution masks, these masks convolve with the input image, and continue to slide over the image, manipulating a square of pixels at a time. Sobel operator uses two kernels one estimating the gradient in the x-direction (columns) and the other estimating the gradient in the y-direction. These kernels are shown in Figure 2.3. One is simply the other rotated by 90° [11].



(a) Original image

(b):Image with Sobel Operator

Figure 2.2 Sobel operator

The magnitude of this gradient is given in equation 2.2, and the orientation is giving by equation 2.4. Figure 2.2(a) show images before applying sobel, and Figure 2.2 (b) image after sobel is used.

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

Gx

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

Gy

Figure 2.3 Sobel convolution masks

2.3 Classification

The purpose of this step is to assign input feature vector to one of the K-existing classes based on classification methods. Different methods could be used to perform the classification, nearest neighbor, and neural networks are what were applied in this project.

2.3.1 Nearest neighbor

Nearest neighbor is one of the most fundamental classification techniques, that is used to classify an object into specific classes. This method works as follows first, a database is created of the existed features that represent the known objects, for which we already know what the correct classification should be. Then, when the algorithm is given a new object to classify, first, it features are extracted, then, the algorithm finds the nearest neighbor of all features in the database. This is done by calculating the Euclidean distance between the new features (test sample) and all the features that reside in the database (training samples), the training samples closest to that test sample are defined as its nearest neighbor. Nearest neighbor gives good and satisfying results when the given database is relatively small, but in a lot of cases when the database is very large, this algorithm fails in giving the best classification results.

Suppose we have N classes, each class is represented by a set of known features z, for each observation vector x, a nearest neighbor list d(x,z) is made for all classes, where d(x,z) uses Euclidean distance.

$$d(x,z)=\sqrt{\sum_{i=0}^{N}(xi-zi)^2} \qquad \textbf{(2.6)}$$

2.3.2 Neural Networks

This section will define neural networks technology as it will one of the approaches used for character recognition step. Neural network layers, training neural networks, and feed forward with back propagation, are what will be discussed in the following subsections.

2.3.2.1 Artificial Neural Network

Artificial neural networks (ANNs) are massively parallel computing systems consisting of a large number of simple processors (neurons) with many interconnections between them [12]. ANNs design were inspired by the way our biological nervous system works and the goal behind the design is to have an "intelligent machines" to solve complex problems such as pattern recognition. ANNs can be viewed as weighted directed graphs in which nodes are artificial neurons and directed edges are connections from the outputs of the neurons to the inputs of the neurons Figure 2.4 shows a simple single layer network. ANNs can be grouped into two classifications: feed forward neural networks in which no loop exists in the graph, and feed back networks where loop exists in the graph.

2.3.2.2 Why ANN's?

There are a wide variety of applications in which neural networks are being applied, such as in medical areas, or machine vision. These networks were developed to solve problems that are difficult or impossible to be solved using the traditional approaches. Pattern recognition is good example of the success of these networks in the field of computer vision.

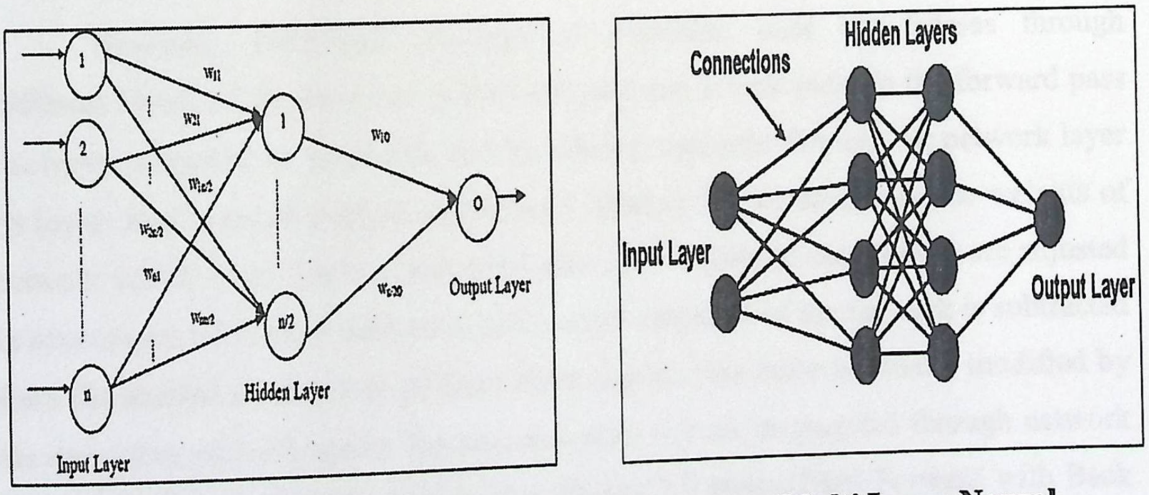
ANNs have several advantages that make it more preferable to use. The most impressive one is adaptive learning; that is the ability to learn how to do tasks based on the data given. Speed, parallelism and ease of implementation are also most often cited as reasons for using a neural network.

2.3.2.3 Network Layers

The most familiar type of neural network consists of three units: input unit which is connected to hidden unit, and hidden unit is connected to output unit see Figure 2.4.

1. First input unit represents raw information that is fed to the network and need to be recognized.
2. The hidden layer takes input unit and the weights on the connection between the input unit and hidden unit.
3. The behavior of output unit depends on the activity of hidden unit and the weights between hidden unit and output unit.

We also differentiate between single layer and multi-layer, in single layer all units are connected to one another Figure 2.4 shows a single layer and multi-layer network.



(a): Single Weighted Feed Forward Neural Layer

(b): Multi Layer Neural Network

Figure 2.4 Single and multilayer networks

2.3.2.4 Training a Neural Network

Neural networks can be explicitly programmed to perform a task by manually creating the topology and then setting the weights of each link, many methods were developed to train this network so as to achieve the required solution of the problem. The next section discusses the basic method for training neural network which is, the feed forward with back propagation

2.3.2.4.1 Feed forward with back propagation

Back propagation algorithm is the most popular algorithm used for training the complex and multilayered networks. A multilayered network consists of input layer, one or more hidden layers and an output layer. Input signal propagates through the network in a forward direction on a layer by layer basis. Training this multilayer is done using error-back propagation algorithm.

Basically, error-back propagation algorithm uses two passes through different layers of the network: A forward pass and a back pass. In the forward pass the input is applied to the nodes, and its effect propagates through the network layer by layer. And a set of outputs is produced. During the forward pass, the weights of network are all fixed. During backward pass, the weights of the network are adjusted in accordance with error correction rule; actual response of the network is subtracted from the desired response to produce error signal. This error is usually modified by the derivative of the transfer function and then is back propagated through network against the direction of the connections. Figure 2.5 shows Feed Forward with Back Propagation.

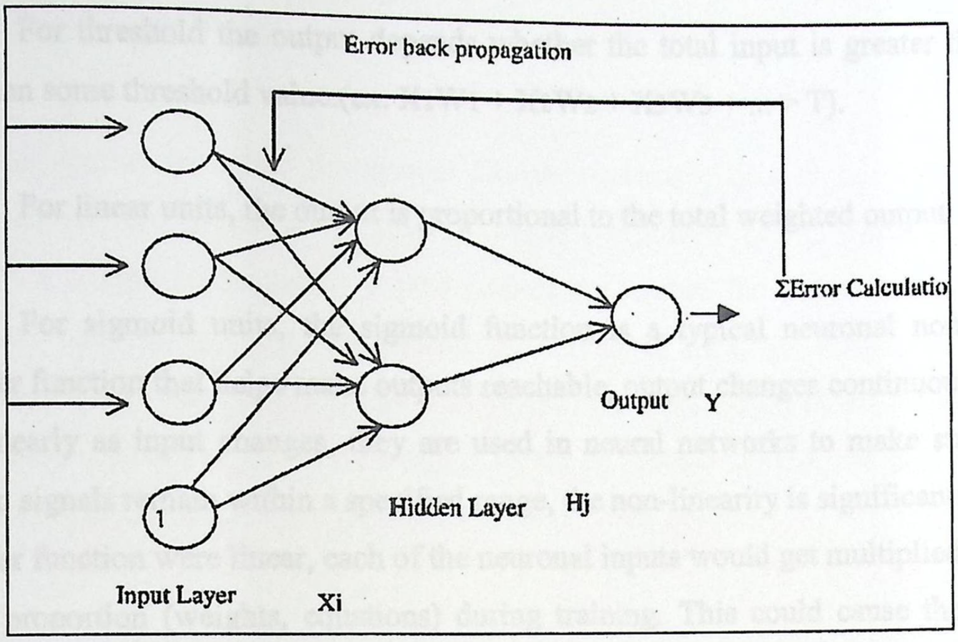


Figure 2.5 Feed Forward with back propagation

So training a three layer network to perform a specific task using back propagation is summed up in the following steps:

1. Choose training pair and copy it to input layer.
2. Cycle that pattern through the net.
3. Calculate error derivative between output activation and target output.
4. Back propagate the summed product of the weights and errors in the output layer to calculate the error on the hidden units.
5. Update weights according to the error on that unit.
6. Repeat until error is low or the net settles.

2.3.2.4.2 Transfer function

ANN's behavior depends heavily on the weights, and the input-output function (transfer function), transfer function falls into three categories: threshold, linear, sigmoid.

For threshold the output depends whether the total input is greater than or less than some threshold value.(ex. $X_1W_1 + X_2W_2 + X_3W_3 + ... > T$).

For linear units, the output is proportional to the total weighted output.

For sigmoid units, the sigmoid function is a typical neuronal non-linear transfer function that helps make outputs reachable, output changes continuously but not linearly as input changes, they are used in neural networks to make sure that certain signals remain within a specified range, the non-linearity is significant. If the transfer function were linear, each of the neuronal inputs would get multiplied by the same proportion (weights, equations) during training. This could cause the entire system to "drift" during training runs. That is, the system may lose outputs it has already tracked while attempting to track new outputs. A non-linearity in the system helps to isolate specific input pathways.

2.4 Hardware Tools

As mentioned earlier the system Design contains photocell that detects the arrival of the vehicle and when it does it sends a signal to the PC which in return initiates a trigger command to a web camera to capture the object to make the necessary analysis to locate the license plate number from the image. Then the system compares the number it has with several numbers stored in the database, If the number matches any it sends a signal to the DC motor to open the gate. In this section we list the necessary components needed in this project.

2.4.1 Sensors

There are two basic types of sensors: analog and digital sensors. The two are quite different in function. An analog sensor produces a continuously varying output value over its range of measurement. For example, a particular photocell might have a resistance of 1k ohm in bright light and a resistance of 300k ohm in complete darkness. Any value between these two is possible depending on the particular light level present. Digital sensors, on the other hand, have only two states, often called "on" and "off." Perhaps the simplest example of a digital sensor is the touch switch. A typical touch switch is an open circuit (infinite resistance) when it is not pressed, and a short circuit (zero resistance) when it is depressed.

2.4.1.1 Photocell

Photocells are made from a compound called cadmium sulfide (CdS) that changes in the resistance when exposed to varying degrees of light. In this project will use Photocells because they are cheaper, easier in implementation, and are available. We use photocells for vehicle detection phase. And when a signal is found it is transferred to the PC. Specialized sensors can be used to senses the vehicle, Unlike Photocell which detects the presence of any object. Figure 2.6 shows a photocell.

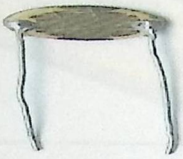


Figure 2.6 Photo cell

2.4.1.2 Magnetic Loop Vehicle Detector

The Magnetic Loop Vehicle Detector (which is the most popular vehicle sensor) is used to detect the presence of a vehicle as it passes over a loop of wire, which has been set into the ground. When the vehicle approached the secured area, this detector starts the cycle by stepping over a magnetic loop detector. The loop detector senses the car and its presence is signaled to the web camera unit. The reason behind using this type of detector is that it only senses the vehicle presence, no other objects will be detected, and so the cam will only capture the image of the vehicle. Figure 2.7 shows a magnetic loop detector.

The Loop Detector can be used for in the following applications:

- As a safety sensor, to prevent a door from closing onto a vehicle.
- To signal a door to open.
- As a vehicle sensor for a car park ticket machine.
- To give a car park barrier a close signal after a vehicle has passed through.

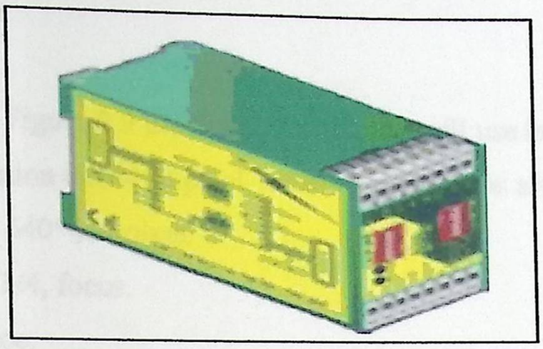


Figure 2.7 Magnetic Loop Vehicle Detector

2.4.2 Dc motor

A DC motor is an electric motor that runs on direct current (DC) electricity .it uses electrical energy to produce mechanical energy. We will use the DC motor in this project to control the gate. Because of their small size and high energy output. See Figure 2.8.

Figure 2.8 DC Motor

2.4.3 Web Camera

As shown in Figure 2.9 the digital camera we will use in this project is a web camera with a resolution 640*480 pixel. Other specifications are listed here:

- Resolutions: 640*480 pixel.
- Sensor size: 1/4, focus.
- USB interface.

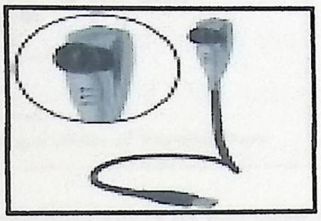


Figure 2.9: Web Camera

2.4.4 Parallel Ports

The Parallel Port is the most commonly used port for interfacing. The port is composed of 4 control lines, 5 status lines and 8 data lines. It's found commonly on the back of the PC as a D-Type 25 Pin female connector. There may also be a D-Type 25 pin male connector. We use the parallel port to interface between the Photocell and the PC. As shown in Figure 2.10.

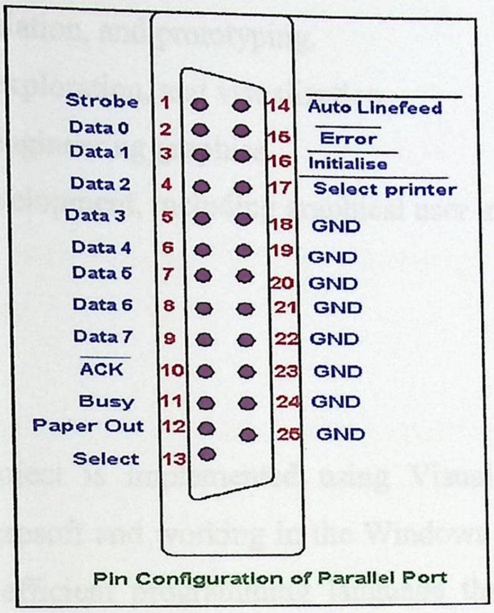


Figure 2.10 Pin configuration of parallel port

2.5 Software tools

Through designing this system matlab will be used as a testing environment in order to test different algorithms before the real implementation is done using visual C++.

2.5.1 MATLAB

MATLAB is a high-performance language for technical computing. It integrates computation, visualization, and Programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation. Typical uses include:

- Math and computation.
- Algorithm development.
- Data acquisition.

- Modeling, simulation, and prototyping.
- Data analysis, exploration, and visualization.
- Scientific and engineering graphics.
- Application development, including graphical user interface building.

2.5.2 Visual C++

The project is implemented using Visual C++ 2005.VC++ is a product of Microsoft and working in the Windows environment ,VC++ is a powerful and efficient programming language that helps in dealing with basic functions in image processing. Opencv libarary was used along with VC++, OpenCV (Open Source Computer Vision) is a library of programming functions mainly aimed at real time computer vision. Example applications of the OpenCV library are Human-Computer Interaction (HCI), Object Identification, Segmentation and Recognition; Face Recognition; Gesture Recognition. The reason behind using this library is to reduce the complexity of the resulting code of VC++, and makes hardware connection easier.

2.6 Summary

This chapter covered the theories, and materials that will be applied through designing this system. Basic concepts of image processing technology were briefly discussed in this chapter.

Neural networks technology, and nearest neighbor were also defined in this chapter. The rest of this chapter mentiond the required software and hardware components needed.

CHAPTER THREE 3

Conceptual Design

3.1 Overview.
3.2 System Structure.
3.3 The Proposed Software Design.
3.4 Summary.

CHAPTER THREE
Conceptual Design

3.1 Overview

This chapter describes the conceptual design of the proposed system, the block diagram of the system is included as well. As mentioned earlier the system software for the identification process would be accomplished by achieving the four major steps which are; image acquisition and pre-processing, plate extraction, segmentation, and character recognition. The techniques that helped in performing these steps would be discussed in the following sections.

This chapter is organized in the following way. The first section shows the block diagram of the whole system, next it discusses the image acquisition using digital camera. The third section describes the extraction plate process, following that segmentation, finally, it discusses the recognition of the segmented characters.

3.2 System Structure

In Figure 3.1, a general block diagram for the overall system is shown. The first block is sensing the arrival of the vehicle using the photocell. When a signal is detected it is transmitted to the connected PC, this causes the PC to trigger a command to the camera to capture the image of the vehicle and this is done in the third block.

After capturing the image, it goes under several levels of analysis such as, image preprocessing in which the image is converted into grey, followed by several techniques to extract the plate, after the extraction of the plate is completed, segmentation begins. It segments the plate into seven regions (Palestinian plate have seven characters), then those segmented characters are passed to the recognition step which is the last phase in image analysis. All theses phases are done in the PC and managed through specific software. The following step relates the number of the processed plate to a database to find a match which is also done in the PC.

If a match is found then the DC motor is signaled to allow the entrance of the Vehicle by opening the gate, otherwise the DC motor does not open the gate, opening and closing is achieved in the last two blocks.

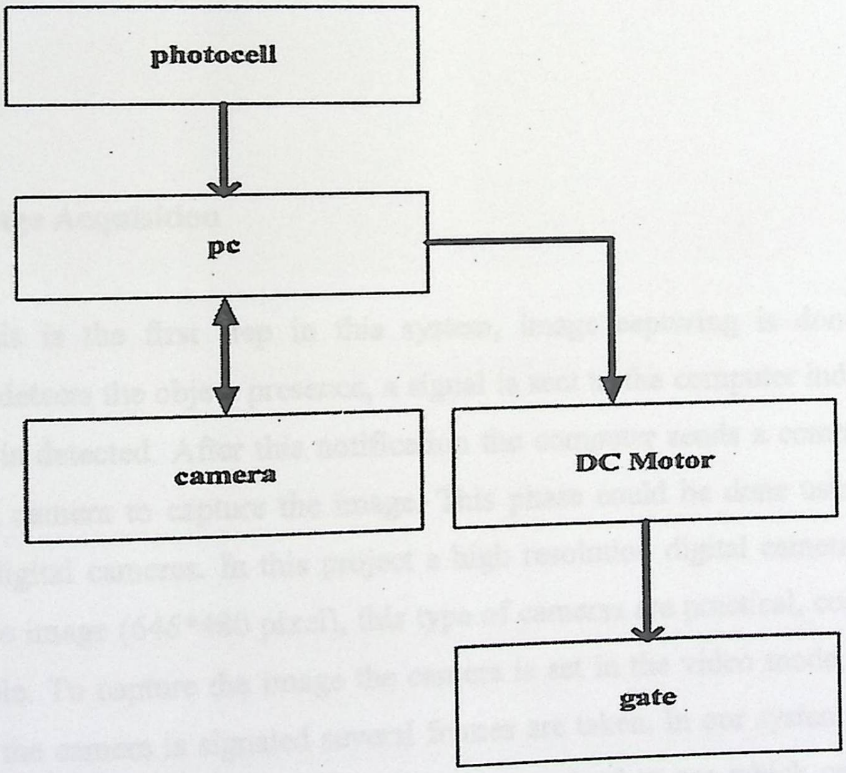


Figure 3.1 System block diagram

3.3 The Proposed Software Design

As mentioned earlier the software of the system deals with four steps, these must be performed correctly in order to achieve the required services. These steps are:

- Image Acquisition and Preprocessing.
- Plate Extraction.
- Segmentation.
- Character Recognition.

In the following sections the algorithm and the techniques that were performed to accomplish these steps are discussed in details. These steps are shown in figure 3.3.

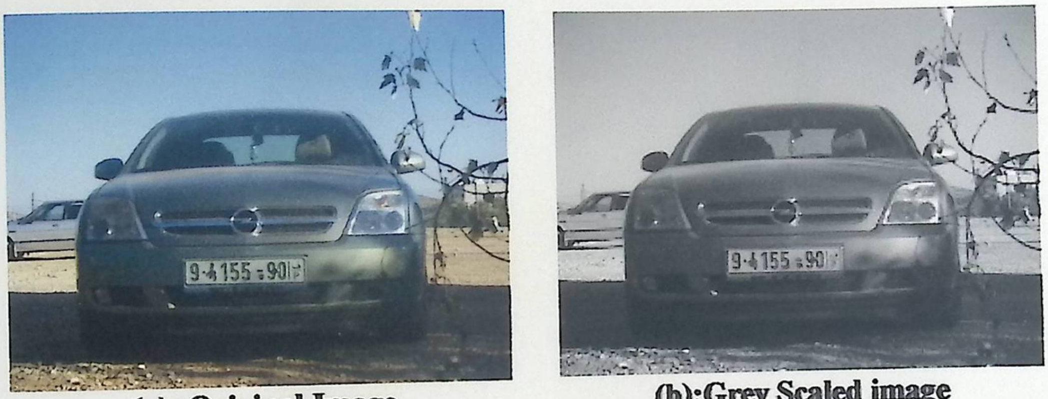
3.3.1 Image Acquisition

This is the first step in this system, image capturing is done after the photocell detects the object presence, a signal is sent to the computer indicating that an object is detected. After this notification the computer sends a command to the connected camera to capture the image. This phase could be done using different types of digital cameras. In this project a high resolution digital camera is used to capture the image (646*480 pixel), this type of cameras are practical, cost effective, and reliable. To capture the image the camera is set in the video mode, this means that once the camera is signaled several frames are taken. In our system the camera takes three frames. When these frames were examined to see which one gives the best image quality, the third frame is selected, and so the code is always set to take the third frame and work the rest of the code on it.

3.3.1.1 Image Preprocessing

Images taken from camera are first preprocessed. The reason behind this is to enrich the edge features to simplify the extraction process. One way is to convert the input image to a gray scale image. The reason behind this is that color is unnecessary information for edges. Figure 3.2 show an image before processing (colored image), and its corresponding grey image. Color (RGB) images are converted into a grayscale image using the following equation [18]:

$$\text{Pixel intensity}= 0.2989*R+0.5870*G+0.1140*B \tag{3.1}$$



(a): Original Image (Before Processing)

(b):Grey Scaled image (After Processing)

Figure 3.2 Image preprocessing

3.3.2 Plate Extraction

Plate extraction is an essential step in the design of this system, as it plays a big role in affecting the overall system accuracy. This step aims to locate the position of the license plate, the idea behind extraction depends heavily on detecting the boundary of the plate because the image usually contains huge data.

This section discusses the phases that were applied in locating the plate region, those phases at first try to decrease the amount of data in the image and avoid the unwanted areas to extract the area which have the same properties of the license plate. The proposed algorithm phases are as follows: First detecting the vertical edge. Then filtering the image to remove the unwanted objects. Next we apply the size and shape to extract those regions that are candidates of being license plate region, and finally vertical edge matching [7], is done in which the license plate is fully extracted and ready to be processed for the following steps. These phases are described in the subsections, and shown in Figure 3.3.

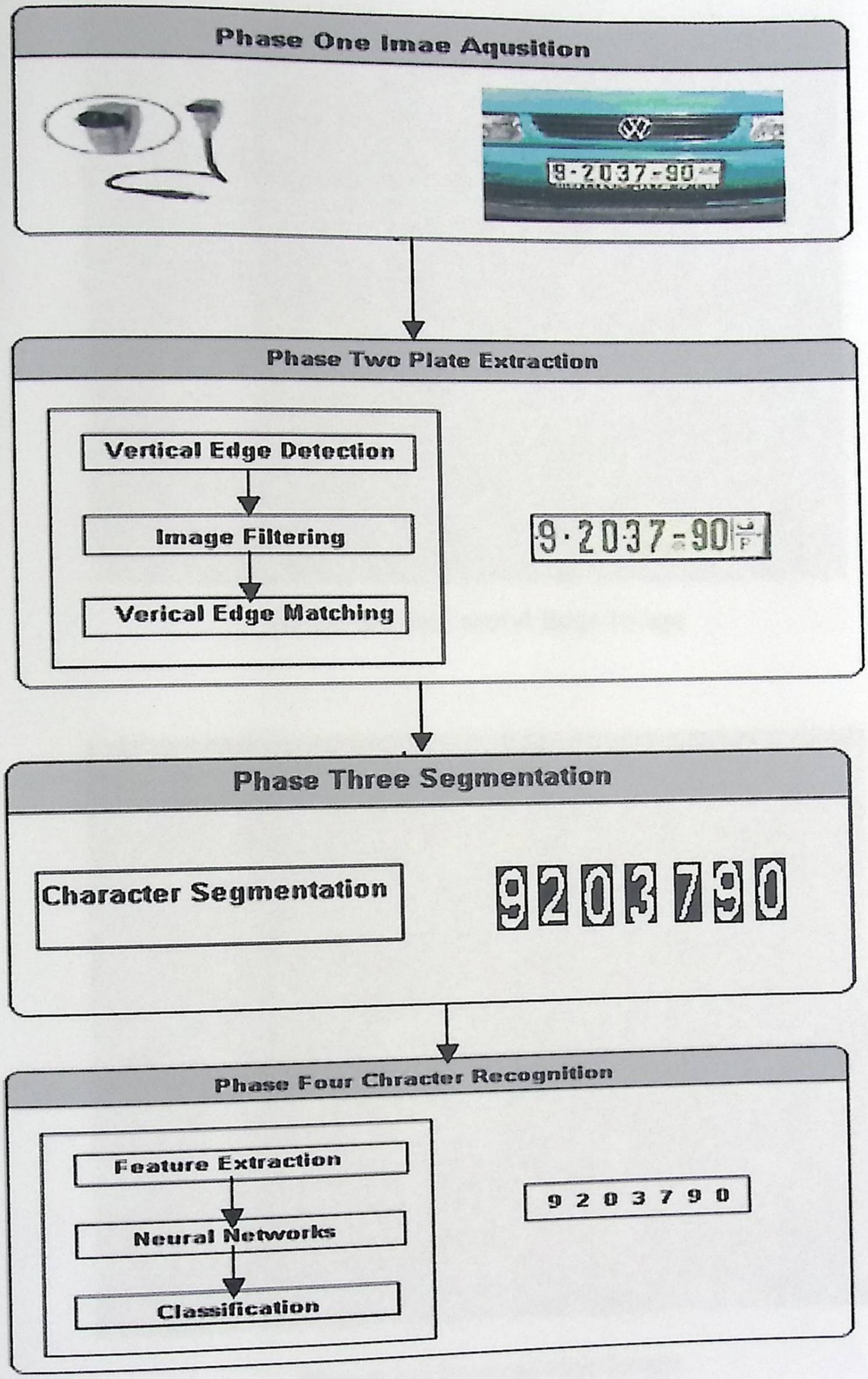


Figure 3.3 Software Design

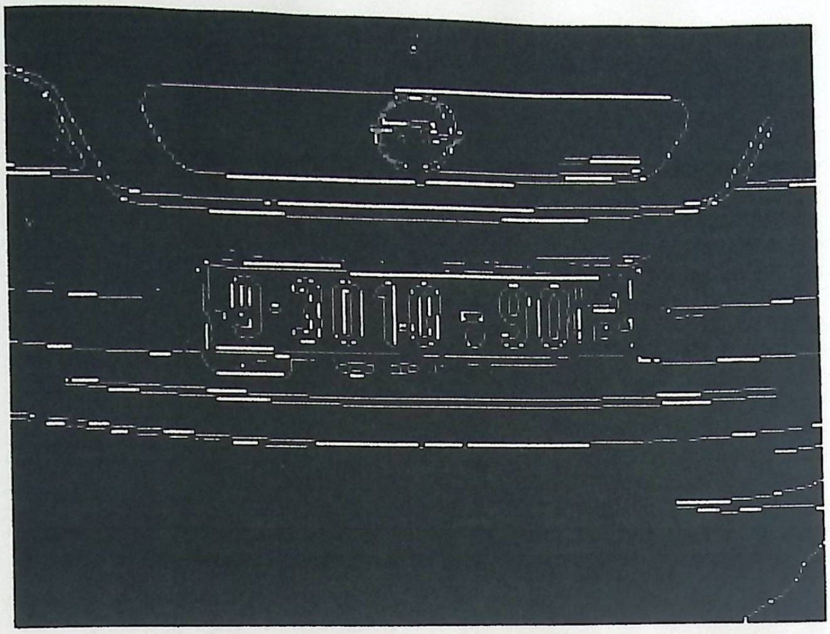


Figure 3.4 Horizontal Edge Image

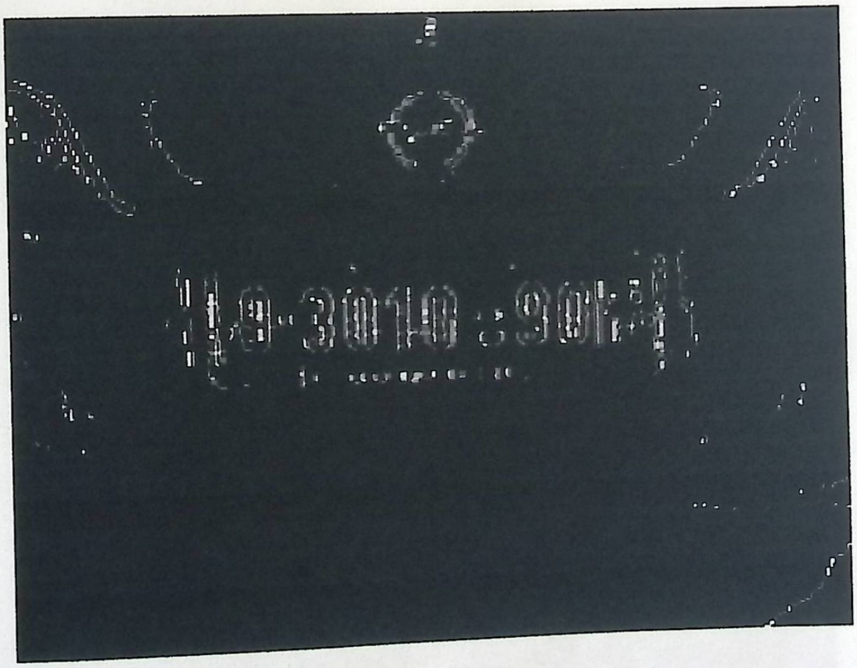


Figure 3.5 Vertical edge Image

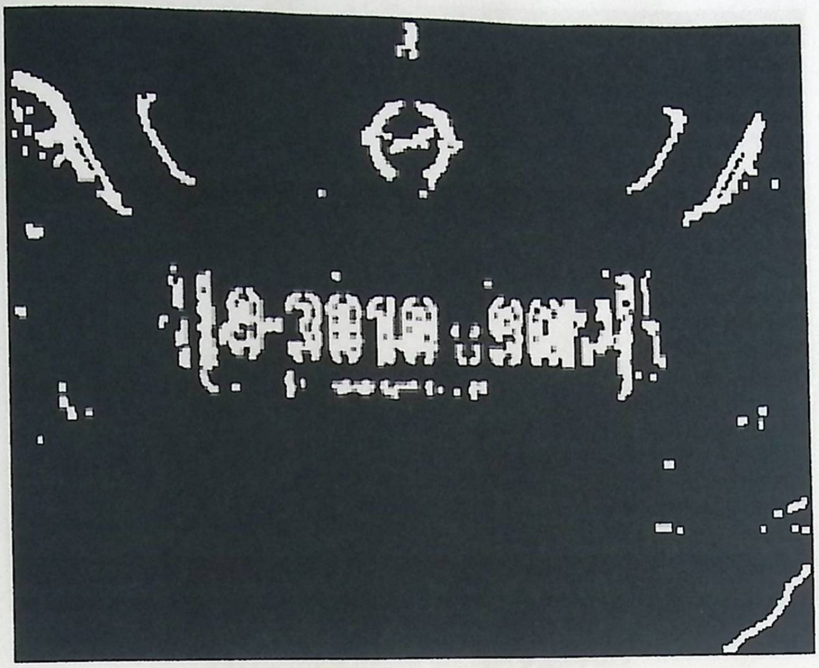


Figure 3.6 Image after Filtering

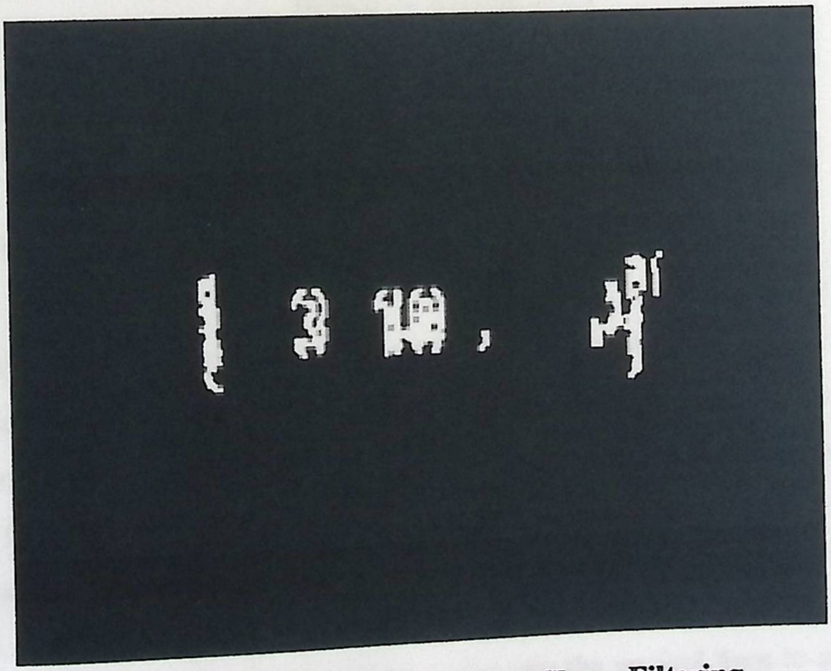


Figure 3.7 Image after size and Shape Filtering

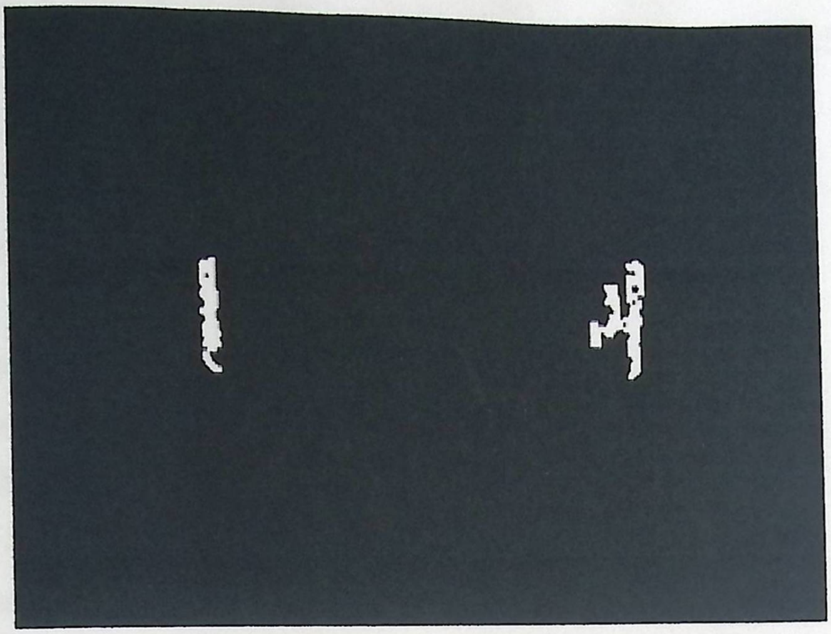


Figure 3.8 Final Image Filtering

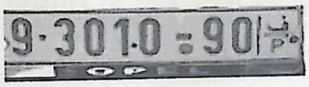


Figure 3.9 License Plate Region

3.3.2.1 Vertical Edge Detection

After converting the image into grey-scale image, its edges are located using sobel operator. See (section 2.2.4.1). Edges as in Figure 3.4 shows that there are more horizontal lines than vertical lines in the vehicle as shown in Figure 3.5. To simplify the extraction method we are only interested in the vertical edges. If the two vertical edges are located correctly, the four corners of the license plate can be detected. As mentioned earlier the sobel operator uses a 3x3 mask, and applying the vertical sobel on the input image would cause only the vertical lines to appear, this edge detection algorithm is not time consuming, and reliable. After applying sobel operator, thresholding is performed to detect the strong edges. Figure 3.5 shows an image after applying the Sobel operator and thresholding.

3.3.2.2 Image Filtering

Filtering is basically used to remove the unwanted objects that don't follow a certain standard. The filtering process goal is to find the regions that are considered to be possible license plate boundaries and discard others by converting them to black. This can be done as follows: first detecting regions, those regions are marked by giving each one a different label. Following this for each region the number of pixels has is counted and for those regions whose pixel's count is less than thirty pixels convert them to black since these are found to be very small lines and are not even close to be a candidate. The results of this step are shown in figure 3.6 and 3.7.

The filtering process is continued to reduce the number of the regions that may be a possible license plate region. To do this any region must be a straight line and follows a specific length. To know if a certain line is straight the slope is calculated using equation 3.2. The first pixel of the region is Po and the last is Pend. We take the slope of the line, as well as the length of the same region, the length of the region is the distance between Po and Pend. If its slope falls in the specified range, and the distance falls within the thresholded value then these regions are now candidates of plate regions. Otherwise regions would be discarded by converting them to black. Figure 3.7 shows image after applying slope and distance filtering.

The slope m of the line through the points (x_1, y_1) and (x_2, y_2) is given by[16]:

$$m = \frac{y_2 - y_1}{x_2 - x_1}, \text{where } (x_1 \neq x_2) \qquad (3.2)$$

The distance between any two points (x_1, y_1), (x_2, y_2) is given by the formula[17]:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \qquad (3.3)$$

3.3.2.3 Vertical Edge Matching

This phase is used to extract the correct plate region, to do this, first the horizontal was calculated distance for each region in respect to the other regions, (ex.Po for region 1with Po's for all the other regions, Pend for region 1with Pend' all other regions), distance is calculated using equation 3.3. Then we applied width to height ratio, the standard ratio of the Palestinian's plate is (4.8:1) if the width to height ratio falls under the specified ratio then these are the boundaries of the license plate. In some cases there is a possibility of more than one pair of region satisfying these conditions, to make sure that the right boundaries are selected the same Yaxe coordinates is taken for both Po, and Pend of the regions. Figure 3.8 shows plate regions, and Figure 3.9 shows the extracted region plate.

3.3.3 Segmentation

Segmentation is the process of partitioning an image into group of pixels which are homogeneous with respect to some criterion. Different groups must not intersect each other, these groups are called segments. Segmentation is usually done to simplify the representation of the image by dividing it into small images that are easier in handling and in analysis.

License plate segmentation or character isolation is used to divide the extracted plate into individual characters. This step is basically done to make the next phase which is character recognition easier in processing. In this step we divide the license plate into seven isolated blocks each containing a single character. This is done because the Palestinian license plate contains seven characters (numbers 0-9).

Different approaches have been used in order to segment the image into sub-images in this project horizontal and vertical projection profile is applied [13].

3.3.3.1 Horizontal and Vertical Projection Profile

Segmentation is done using the horizontal and vertical projections, the method first find the horizontal profile of the binarized license plate, then the vertical projection is performed on the resultant image obtained after horizontal projection, to segment the plate into seven individual characters.

Horizontal projection of the thresholded plate often reveals the exact character positions, this is done by counting the number of black pixels for all columns corresponding to a particular row. The obtained count is then plotted to obtain the resultant projection for that particular row. This process continues by repeating it for all the rows to get the overall projection profile. Horizontal projection is done at first to remove the frame of the plate and any other unwanted pixels obtained due to dust and scratches on the plate. At the end of this process it is expected to have the plate containing only the characters. This is shown in figure 3.12.

Vertical Projection is obtained after horizontal projection is done, that means that the result of horizontal phase is the input to vertical projection step. Vertical projection is done by finding the number of black pixels for all the rows corresponding to each column is found. The obtained count is plotted to get the vertical projection for a particular column. This process is repeated for all the columns and the resultant plot gives the over all projection profile.

After this phase it is expected to have the license plate segmented into seven blocks, each block contains a single character, characters are segmented depending on the transitions from a group of black pixels to a white pixel. Figure 3.10 shows these steps. And figure 3.11 shows the final result of segmentation.

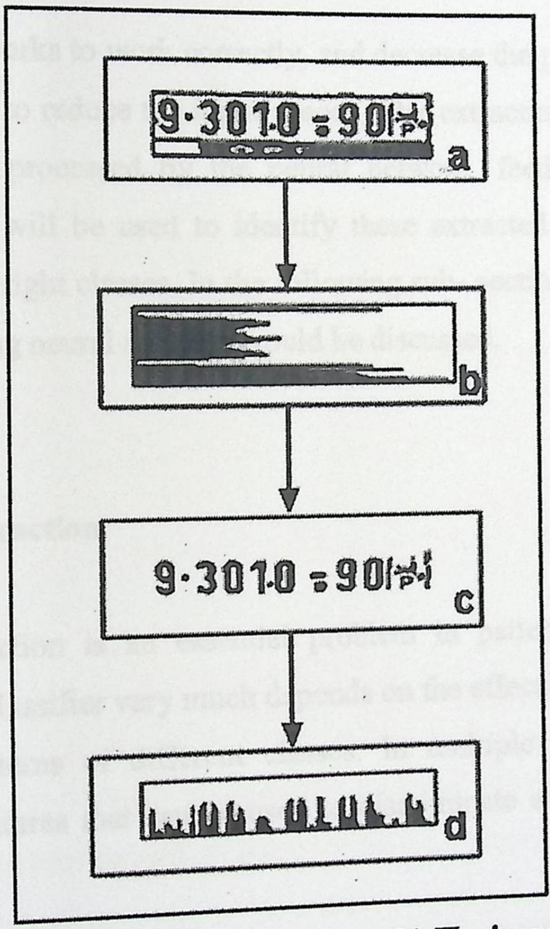


Figure 3.10 (a) Extracted plate, (b) Horizontal projection, (c) plate with frame removed, (d) Vertical projection.

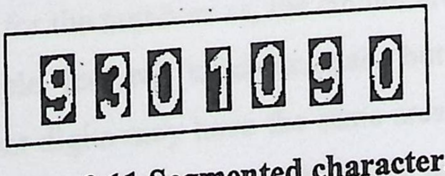


Figure 3.11 Segmented characters

vertical projection for a particular column. This process is repeated for all the columns and the resultant plot gives the over all projection profile.

After this phase it is expected to have the license plate segmented into seven blocks, each block contains a single character, characters are segmented depending on the transitions from a group of black pixels to a white pixel. Figure 3.10 shows these steps. And figure 3.11 shows the final result of segmentation.

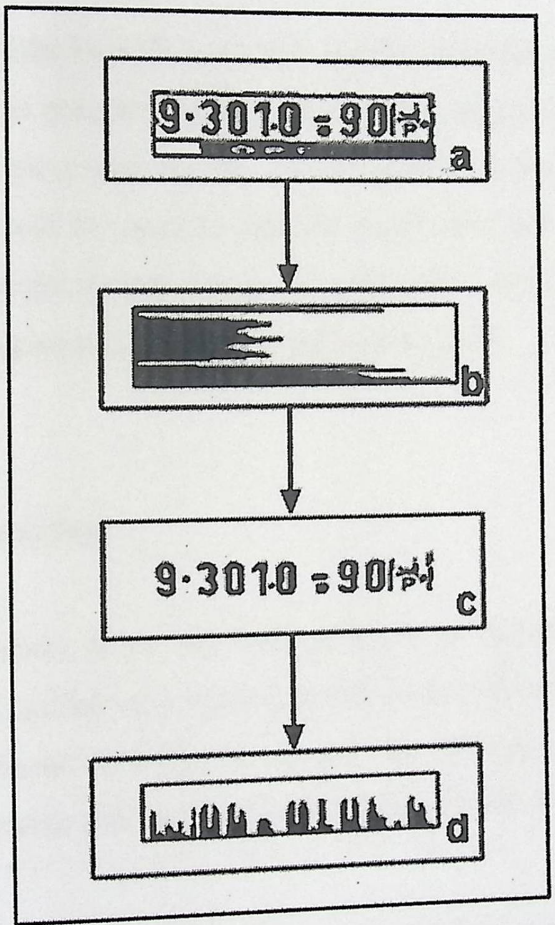


Figure 3.10 (a) Extracted plate, (b) Horizontal projection, (c) plate with frame removed, (d) Vertical projection.

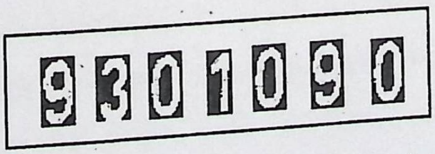


Figure 3.11 Segmented characters

3.3.4 Character Recognition

Character recognition is the final step in designing this system. This step deals with the recognition of the characters isolated in the phase of character segmentation, as they represent the input to this step. Character recognition will be achieved by training neural networks with a good and sufficient dataset of input that makes these networks capable of recognizing these characters.

For these networks to work correctly, and decrease the pressure on it, feature extraction is required to reduce the input space. After extracting these features they will be ready to be processed by the neural network, feed forward with back propagation network will be used to identify these extracted features in order to classify them to their right classes. In the following sub- sections feature extraction, and classification using neural network would be discussed.

3.3.4.1 Feature Extraction

Feature extraction is an essential problem in pattern classification. The success of a pattern classifier very much depends on the effectiveness of the features representing the patterns of different classes. In multiple pattern classes, it is important to find features that can be used to discriminate each class from all the other classes.

Feature selection is the first step in this process. One should be able to select the most representative and efficient features in the image, in the plate image several features could be selected for the numbers as, the circumference and area of the digit as they present the simple features to distinguish, but these features are not sufficient, because different digits may have the same area and circumference as in the case of '6' and '9'. So additional features should be added to ensure theses are distinguished correctly such as, the number of endpoints in the upper half and lower half, to explain more digit '6' has one endpoint in the upper half and zero endpoints in the lower half, while the endpoints of digit'9' are the reverse of digit '6'. One could also choose the horizontal and vertical projections of the character as mentioned earlier, or the edges of each character.

After the experimental tests that were made to choose the best features, the results gave a high recognition rate when horizontal projections, and the edges were applied.

3.3.4.2 Recognition and Classification

After features are extracted they are stored in patterns. Pattern is a quantitative description of an object. This pattern is usually arranged in the form of feature vector.

$$X= \begin{bmatrix} X1 \\ X2 \\ . \\ . \\ . \\ Xn \end{bmatrix}$$

Where X1, X2, ... Xn are the selected features. Where the class is a set of patterns that share common attributes [14].

Then these patterns are ready to be processed and classified. As mentioned earlier neural approach will be used for this purpose. In this project feed forward with back propagation technique is applied for classification. This method has been performed successfully in pattern recognition and became famous and reliable method for recognizing characters. These steps are summarized here, but for more information see section 2.3.2.4.1.

First you enter the pattern to input layer, then this pattern is cycled through the network, and for each cycle the error derivative is calculated between actual response and the desired response. Then this error is back propagated through the network and the weights are adjusted to make actual response of the network move closer to the desired response. In other words until error is low or the net settles.

In the recognition phase the neural network is implemented using three layers, input layer, hidden layer, and output layer. The number of neurons in the input layer depends on the feature vector size, which was found to be 70 neurons. The number of neurons in the hidden layer was calculated according to equation 3.3 [15] which gave a result of 28 neurons. Finally, the output layer neurons are 11 outputs, ten of them for characters 0-9, and the last neuron is active in cases when the system fails in the recognition.

$$\text{\#of neurons in hidden layer}=\sqrt{(\text{\#input neurons X \#ofoutput neurons}} \qquad (3.3)$$

3.4 Summary

This chapter discussed the conceptual model of the License plate recognition system. The implementation issues related to the four stages: Image acquisition, License plate extraction, segmentation and recognition were discussed in details. For the extraction phase, three steps were followed: vertical edge detection, image filtering, finally, vertical edge matching. For segmentation horizontal and vertical projections is applied, and for recognition neural approach is performed.

CHAPTER FOUR 4

Implementatiom

CHAPTER FOUR
Implementation

4.1 Overview

This chapter at first describes how the whole system is connected, then the pseudo code for software phases are presented.

4.2 System Model

A model for the system is designed, in this model we made our best that it would simulate the real world. Special license plates were designed for this purpose, that is similar to the real ones. Photocell is placed within a range of 25cm-35 cm away from the camera position. A green led is used to indicate the opening process, and a red led is used to indicate a failure.

4.3 Detailed Description of the Program Phases

The system goes through three main phases: the input phase, the processing phase, and the output. These phases are explained as the following:

4.3.1 Input phase

This system has one input that comes from the photocell device which is connected to the computer through parallel port. After the Photocell detects the arrival of the vehicle it sends a signal to the connected computer for the web camera to capture the image of the object.

4.3.2 Processing phase

The processing phase of this system is done after the analysis process of the vehicle image is completed. After the system makes the necessary analysis to locate the plate from the image and identifies the numbers, a comparison is made between the newly extracted plate number, and the other plates numbers stored in the system database and examine if there is a match between them, the results of the comparison step determines what kind of action must the system do.

4.3.3 Output phase

The output of this system is divided into two actions, these actions are:

a. if a match is found:

- Green led will light, this is used to indicate the opening of the gate. As shown in figure 4.1.
- A welcome message is given to the driver.

b. if no match is found:

- The green led will stay off, this number is not authorized to access in.
- The system will give a message to the driver (Can not access You are not registered here).

There are some cases where the system may fail such as:

1. A failure in image acquisition phase, the camera couldn't capture the object.

2. A failure in the extraction phase, at this case the system has extracted a different object, or couldn't extract anything. This will lead to false segmentation, and at the recognition phase the unknown neuron to give an active state.
3. A failure in the recognition phase, the system couldn't identify the characters correctly, which will also lead to unknown neuron to give an active state.

At these cases a red led is lightened to indicate a failure, and another image is taken and analyzed, as shown in figure 4.1.

4.4 Interfacing circuit

Parallel port is a very commonly known port, if you see backside of your computer, there will be a port having 25 pins. The PC's Parallel port had a total of 12 digital outputs and 5 digital inputs, the remaining 8 pins are grounded. To see this circuit see figure 4.2.

In this project four pins were used, to see the number of used pins see table 4.1.

D25-pin Number	Direction	Function
2	In/Out	Connect it to red LED to show driver there is problem.
6	In/Out	Connect to green LED to show driver you can access and gate open.
12	In	Connect to photocell to detect the arrival of the vehicle.
25	Gnd	Ground.

Table 4.1 used parallel port Pins

To program the parallel port in VC++ and using OS windows XP Inpout32.dll is used. InpOut32 is a DLL file which can send a data to parallel port, and receive the data from the parallel port.

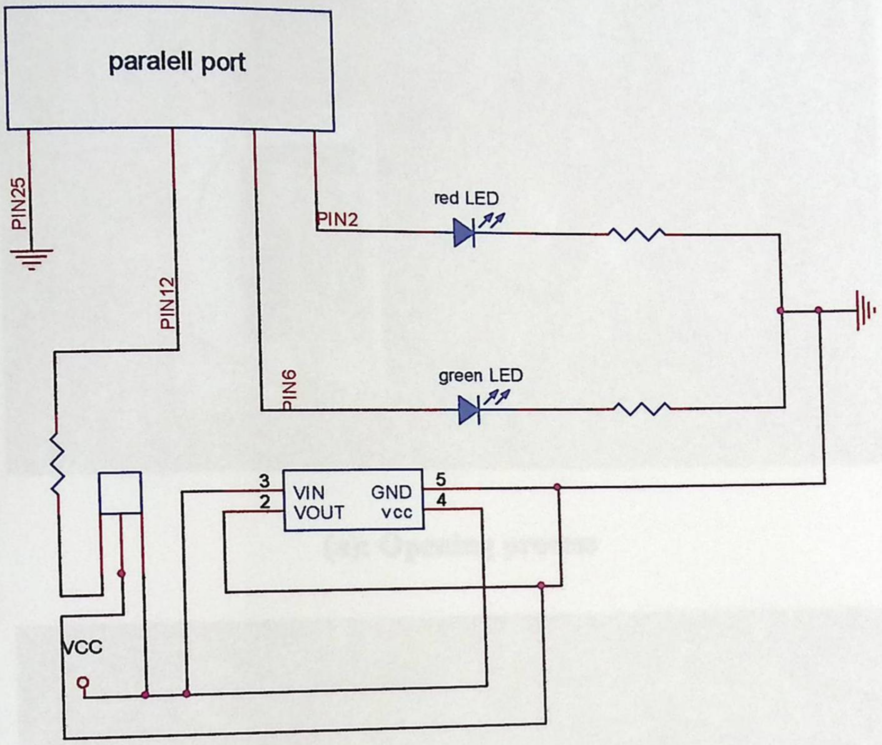


Figure 4.2 interfacing circuit

Positive voltage regulator(L7805C-V):

As one know that the microcontroller in the computer works at 5 v, while the used photo- cell operates on 12 v, and because of this a signal cant be detected from it immediately, so a positive voltage regulator is used to convert the voltage into 5v. See figure 4.3.

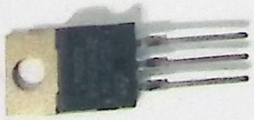
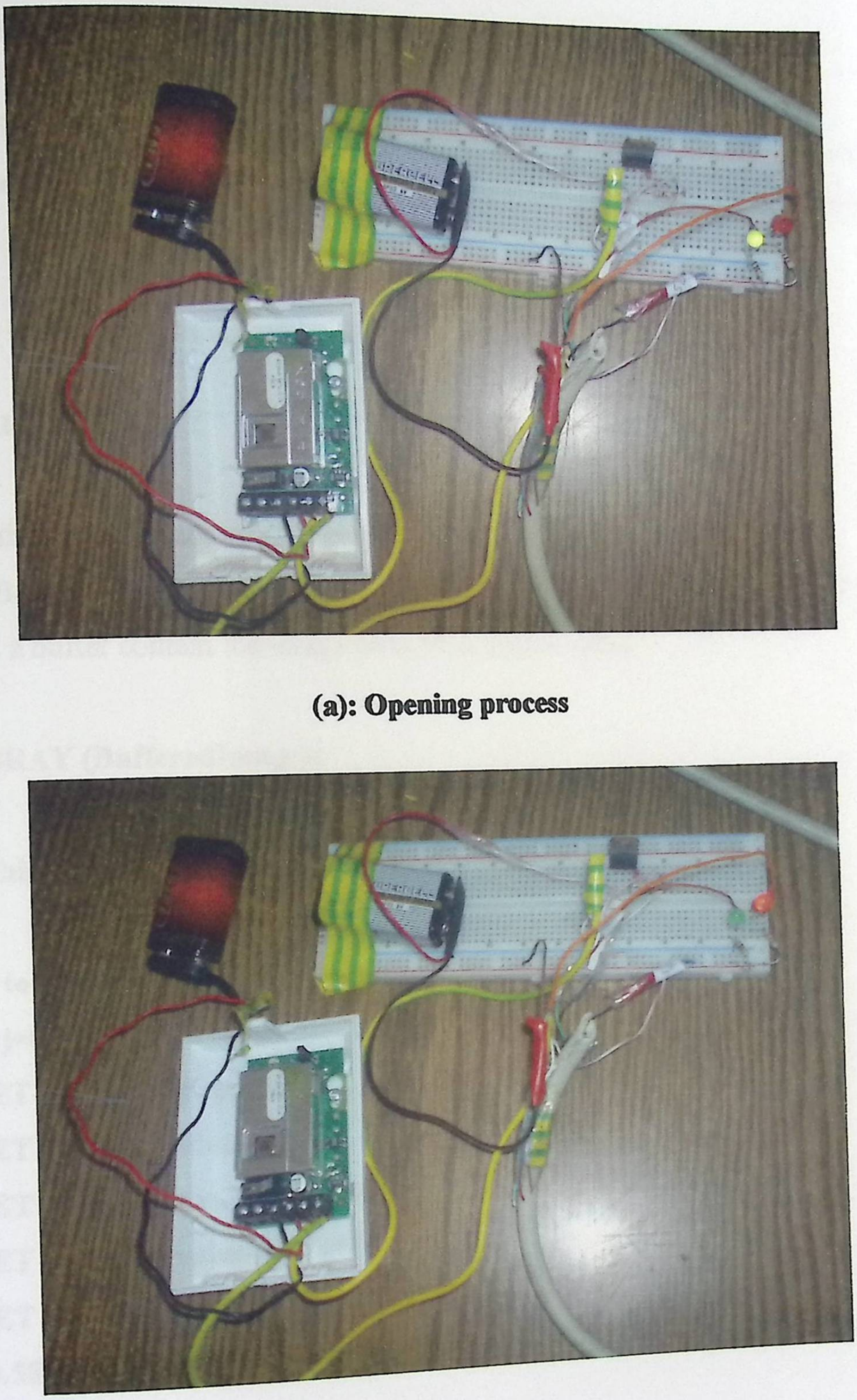


Figure 4.3 positive voltage regulator



(a): Opening process

(b) :A Failure detection

Figure 4.1 Output cases

4.5 Pseudocode

This section describes the main functions used in programming the code in the phases, it lists their pseudeocode as away of showing how these functions worked.

List 4.1: Extraction phase

List 4.1.1: RGB2GRAY

Description: convert colored image pixels from RGB to gray scale .

Input: BufferedImage – a buffer contains the colored image to be converted .

Output: a buffer contain the image converted pixels value .

```
RGB2GRAY (BufferedImage){

Image (hight , width )

For i=0 to BufferedImage hight
    For j=0 to BufferedImage width
        SET Grayimage position (i, j) to 1
        SET r to Red value of pixel
        SET g to Green value of pixel
        SET b to Blue value of pixel
        SET Image position (i, j) to ((0.2989 MULTIPLIED by r) ADD to
        (0.5870 MULTIPLIED by g) ADD to (0.1140 MULTIPLIED by b))
         DIVIDED by 255
    End For
End For

Return Image
}
```

List 4.1.2: Convolution

Description: a convolution of any matrix with any 3X3 kernels .

Input: Matrix– a matrix of gray value from the captured image .
Height – Matrix height .
Width – Matrix Width.
Kernel – a matrix of 3X3 that will convolution with Matrix .

Output: a matrix contains the result of the convolution .

```
Convolution (Matrix , Height , Width , Kernel ){

Rmatrix (Height -2 , Width -2)

For i=1 to Height -1
  For j=1 to Width – 1
   SET Rmatrix position (i-1, j-1) to
           Matrix position(i-1 , j-1 ) MULTIPLID by Kernel position(0) ADD to
           Matrix position(i-1 ,  j   ) MULTIPLID by Kernel position(1) ADD to
           Matrix position(i-1 , j+1) MULTIPLID by Kernel position(2) ADD to
           Matrix position(  i  , j-1 ) MULTIPLID by Kernel position(3) ADD to
           Matrix position(  i  , j   ) MULTIPLID by Kernel position(4)  ADD to
           Matrix position(i-1 , j+1) MULTIPLID by Kernel position(5) ADD to
           Matrix position(i+1 , j-1) MULTIPLID by Kernel position(6)  ADD to
           Matrix position(i+1 ,  j  ) MULTIPLID by Kernel position(7)  ADD to
           Matrix position(i+1, j+1) MULTIPLID by Kernel position(8)

  End For
End For

Return Rmatrix
}
```

List 4.1.3: Labeling

Description: Convert binary image into labeled regions.

Input:binaryimage- abuffer contain gray image .

Output: abuffer contain a binary image with the regions being labled.

```
Labeling(binaryimage) {

Matrix(hight , width)

For  i=0 to  binaryimage hight
   For  j=0 to  binaryimage width .
     IF  binaryimage position (i, j) is NOT EQUAL  zero THEN
     SET  binaryimage position (i, j)  to 1
     End IF
  End For
End For

SET count to 10

For  i=0 to  binaryimage hight
   For  j=0 to  binaryimage width .
     IF binaryimage position (i, j) is NOT EQUAL  zero THEN
       IF i-1 >=0
         IF binaryimage position (i-1, j) >  1  THEN
          SET binaryimage position (i,j) to value of  binaryimage position (i-1,j)
          SET Matrix position (i,j) to 1
         End IF
       End IF
     End IF
```

```
    IF  binaryimage position (i, j) is NOT EQUAL  zero THEN
      IF j-1 >=0
         IF binaryimage position (i, j-1) >  1  THEN
           SET binaryimage position (i,j) to value of  binaryimage position (i,j-1)
           SET Matrix position (i,j) to 1
         End IF
      End IF
    End IF

 IF Matrix position (i,j) is NOT EQUAL  1 THEN
   IF i+1 <=  bianryimage hight
    IF binaryimage position (i+1, j) is EQUAL  1  THEN
      IF j-1 >=0
       IF binaryimage position (i+1, j-1)  >  1  THEN
        SET binaryimage position (i, j) to value of binaryimage position(i+1, j-1)
        SET Matrix position (i,j) to 1
        End IF
      End IF
    End IF
   End IF
 End IF

 IF Matrix position (i,j) is NOT EQUAL  1 THEN
   SET binaryimage position (i, j) to count
   SET Matrix position (i,j) to 1
   INCREMENT count
 End IF


    End For
 End For
```

```
      IF  binaryimage position (i, j) is NOT EQUAL  zero THEN
        IF j-1 >=0
          IF binaryimage position (i, j-1) >  1  THEN
            SET binaryimage position (i,j) to value of  binaryimage position (i,j-1)
            SET Matrix position (i,j) to 1
          End IF
        End IF
      End IF

    IF Matrix position (i,j) is NOT EQUAL  1 THEN
      IF i+1 <=  bianryimage hight
        IF binaryimage position (i+1, j) is EQUAL  1  THEN
          IF j-1 >=0
            IF binaryimage position (i+1, j-1)  >  1  THEN
              SET binaryimage position  (i, j) to value of binaryimage position(i+1, j-1)
              SET Matrix position (i,j) to 1
            End IF
          End IF
        End IF
      End IF
    End IF
  End IF
```

```
Return binaryimage
}
```

List 4.1.4: Count the number of pixels for each region

Description: calculate number of pixel of each connected objects .

Input: Image – a buffer contains labeling image .

count – number of connected objects .

Output : a matrix contain the number of pixel of each connected objects .

```
Num_of_pixel_object (Image , count ){

Matrix( hight , width )

For i=0 to Image hight
    For j=0 to Image width
      For k=10 to count
         IF image position (i, j) is EQUAL k THEN
           INCREMENT Matrix position (1, k) to 1
         End IF
      End For
    End For
 End For

 Return Matrix
 }
```

List 4.1.5: Finding coordinates

Description: determine the y-x axis of the starting connected object and the ending.

Input: Image – a buffer contain labeling image .

count – number of connected objects .

Matrix – contain the number of pixel of each connected objects .

Output: Matrix1 contain y-x axis the starting connected object and the ending.

```
Coordinate(Image , count , Matrix){

Matrix1(height , width)

For i=0 to Image height
  For j=0 to Image width
    For i=10 to count
      IF Image position (i, j) is EQUAL k THEN
        IF Matrix1 position (5, k) is EQUAL 0 THEN
              SET Matrix1 position (5, k) to 1
              SET Matrix1 position (1, k) to i
              SET Matrix1 position (2, k) to j
        ELSE
              INCREMENT Matrix1 position (6, i) to 1
        End IF

        IF Matrix1 position (6, k) is EQUAL to value of Matrix position (1, k)
            THEN
              SET Matrix1 position (3, k) to i
              SET Matrix1 position (4, k) to j
            End IF
        End IF
      End For
    End For
  End For

  Return Matrix1
  }
```

List 4.1.6: Finding slope and length

Description: calculate the slope and length of each object .

Input: Matrix – contain y-x axis the starting object and the ending.

count – number of object .

Output: Matrix1 contain slope and length of each object.

```
Slope(Matrix , count){

Matrix1(height , width)

 For i=10  to count
     SET S to abs(SUBTRACT  Matrix position (4,i) from   Matrix position (2,i))
     SET k to abs(SUBTRACT  Matrix position (3,i) from   Matrix position (1,i))
     SET Matrix1 position (1,i) to (S DIVIDED by k)
     SET Matrix1 position (2,i) to(sqr(pow(S,2) add to pow(K,2)))
 End For

 Return Matrix1
 }
```

List 4.1.7: Finding Ratio

Description: calculate the ratio of each object in respect to all other objects.

Input: Matrix1– contain y-x axis the starting object and the ending.

Matrix2– contain length of each object .

count – number of object .

Output: Matrix3 contain ratio of each object to others .

```
Ratio(Matrix1, Matrix2 , count){

Matix3(hight , width)

For i=10 to count
  For j=10 to count
    SET y to abs(SUBTRACT Matrix1 position (1,i) from Matrix1 position (1,j))
    SET w to abs(SUBTRACT Matrix1 position (2,i) from Matrix1 position (2,j))
    SET P to(sqrt(pow(y,2) add to pow(w,2)))
    SET Matix3 position (i,j) to (p DIVIDED by Matix2 position (2,i))
  End For
End For

Return Matix3
}
```

List 4.1.8: Finding vertical two lines of plate

Description: determine vertical two lines of plate .

Input: Matrix1– contains slope of each object .

Matrix2– contains ratio of each object in respect to all other objects .

count – number of object .

Output: Matrix3 contains target two lines of plate.

```
Two-Line_Detection(Matrix1, Matrix2 , count){

Matrix3 ( hight , width)
For i=10  to count
  For j=10  to count
    IF Matrix1 position (1,i) is EQUAL or LESS than the desired rang AND
       Matrix1 position (1,j) is EQUAL or LESS than the desired rang AND
       Matrix2 position (i,j) is in the desired rang THEN
       SET Matrix3 position (i,j) to 1
    End IF
  End For
End For

Return Matrix3
}
```

List 4.2: Segmentation

List 4.2.1: Horizontal projection

Description: calculate number of pixel of each row of image .

Input: Image– a buffer contains extracted plate .

Output: Matrix contain number of pixel of each row of image .

```
Horizontal_projection(Image){

Matrix(hight , width)
For i=0 to Image hight
   For j=0 to Image width
      IF Image position (i, j) is NOT EQUAL zero THEN
      INCREMENT Matrix position (i) to 1
      End IF
   End For
End For

Return Matrix
}
```

List 4.2.2: Filter_plate

Description: Filter the plate image from unwanted objects.

Input: Image–a buffer contains extracted plate.

Matrix contains number of pixel of each row of image.

Output: a buffer contains image of plate without any noise.

```
Filter_plate (Image, Matrix){

SET min1 to Matrix position (0)
SET min_position1 to 0
SET min2 to Matrix position(Matrix height)
SET min_position2 to Matrix height

For i=1 to Matrix height
 IF i > (Matrix height divided by 2) AND Matrix position(i) LESS THAN min1
 THEN
    SET min1 to Matrix position(i)
    SET min_position1 to i
  End IF
End For

For  i= Matrix height  to  0
  IF i < (Matrix height divided by 2) AND Matrix position(i) LESS THAN
      min2 THEN
      SET  min2 to Matrix position(i)
      SET min_position2 to i
  End IF
End For

For  k=0  to  Image height
  IF  k  EQUAL  min_position1 THEN
     SET all row LESS THAN k in Image with all column to zero
   End IF
  IF  k  EQUAL  min_position2 THEN
     SET all row MORE THAN k in Image with all column to zero
  End IF
End For

Return  Image
}
```

List 4.2.3: Vertical projection

Description: calculate number of pixel of each column in the image.

Input: Image– a buffer contains Filtered extracted plate.

Output: Matrix contain number of pixel of each column of image.

```
Horizontal_projection(Image){

Matrix (height , width )

For i=0 to Image width
  For j=0 to Image height
    IF Image position (i, j) is NOT EQUAL zero THEN
    INCREMENT Matrix position (j) to 1
    End IF
  End For
End For

Return Matrix
}
```

List 4.2.4 Segmentation

Description: Segment plate into several parts.

Input: Image– a buffer contains filtered extracted plate.
       Matrix–contains vertical projection.

Output: Array of three dimensions contains divided part.

```
Segmentaion(Image , Matrix){

SET count to 0
SET t to 1

For i=0 to Image width
  IF Matrix position (i) is EQUAL zero AND count EQUAL zero THEN
    SET s to i
```

```
      SET i to 1
    End IF
  IF Matrix position (i) is EQUAL zero AND count EQUAL 1 THEN
    SET e to i
    SET i to 2
  End IF

  IF count EQUAL 2 THEN
    SET aa to1
    SET bb to1
      For j=0 to Image height
        INCREMENT aa
        SET bb to1
          For y=s to e
            INCREMENT bb
            SET Array position (t,aa,bb) to Image position (j,y)
          End For
        End For
      INCREMENT t
      SET count to zero
    End IF
  End For

  Return Array
  }
```

```
      SET i to 1
    End IF
  IF Matrix position (i) is EQUAL zero AND count EQUAL 1 THEN
    SET e to i
    SET i to 2
  End IF

  IF count EQUAL 2 THEN
     SET aa to1
     SET bb to1
       For j=0 to [illegible] , Bin){
         INCREMENT aa
         SET [illegible] width)
  SET max to Matrix1 position (1)
  SET d to zero
            [illegible]

  For i=2 to Matrix1 height
     IF max < Matrix1 position (i) THEN
        SET max to Matrix1 position (i)
        SET d to i
     End IF
  End For

   IF Bin MODULUS 2 is EQUAL zero THEN
    SET t to Bin DIVIDED by 2
    SET s to (Matrix1 position (t) ADD TO Matrix1 position (t+1)) DIVIDED by 2
    SET r to (Matrix1 position (t+35) ADD TO Matrix1 position (t+36))
              DIVIDED by 2
    SET Matrix1 position (t) to s
    SET Matrix1 position (t+35) to r
```

List 4.3: Recognition

List 4.3.1: features modification

Description: Preparing the Features to enter the neural network.

Input: Matrix1- contain feature extraction of plate, horizontal and edge detection.
Bin – number of feature of the matrix which contains the feature.

Output: Matrix2 contains the final modified features.

```
Feature modification(Matrix1 , Bin){

Matrix2(height , width)
SET max to Matrix1 position (1)
SET d to zero

For i=2 to Matrix1 height
   IF max < Matrix1 position (i) THEN
       SET max to Matrix1 position (i)
       SET d to i
   End IF
 End For

IF Bin MODULUS 2 is EQUAL zero THEN
  SET t to Bin DIVIDED by 2
  SET s to (Matrix1 position (t) ADD TO Matrix1 position (t+1)) DIVIDED by 2
  SET r to (Matrix1 position (t+35) ADD TO Matrix1 position (t+36))
              DIVIDED by 2
  SET Matrix1 position (t) to s
  SET Matrix1 position (t+35) to r
```

```
For j=1 to Bin +1
  IF j NOT EQUAL  t+1 THEN
    SET Matrix1 position (d) to Matrix1 position (j)
    SET Matrix1 position (d+35) to Matrix1 position (j+35)
    INCREMENT d
  End IF
End For

DECREMENT Bin
SET w to  35 SUBTRACT from Bin
SET q to zero

For k=0 to w ADD to Bin
   SET Matrix2 position (k) to Matrix2 position (q) DIVIDED by max
   SET Matrix2 position (k+35) to Matrix2 position (q+35)
   INCREMENT q
End For

ELSE

For j=1 to Bin +1
     SET Matrix1 position (d) to Matrix1 position (j)
     SET Matrix1 position (d+35) to Matrix1 position (j+35)
     INCREMENT d
End For

SET w to  35 SUBTRACT from Bin
SET q to zero

For k=0 to w ADD to Bin
```

```
    SET Matrix2 position (k) to Matrix2 position (q) DIVIDED by max
    SET Matrix2 position (k+35) to Matrix2 position (q+35)
    INCREMENT q
End For

Return Matrix2
}
```

4.6 Summary

This chapter introduced the hardware implementation. The interfacing connection is also mentioned. The pesudocode of the methods used in image analysis, plate extraction, segmentation, and recognition are presented.

CHAPTER FIVE 5

Experimental analysis and results

CHAPTER FIVE

Experimental analysis and results

5.1 Introduction

This chapter describes the that were done on all the work described in the previous chapters. The tests were first performed separately for each phase, then the whole system is tested, the results of these tests are presented, and described in this chapter.

5.2 Project implementation

The project is implemented using Visual C++, and opencv library. The tests that were performed were applied using matlab 7.0.1 at first to see if the results are satisfying and good.

5.3 Extraction test

The proposed method for the extraction phase as mentioned earlier deals with three major steps which are: vertical edge detection, filtering , and vertical edge matching. This section presents the test that was performed to verify the accuracy of these steps and the rate of success for this algorithm.

5.3.1 Test data and test description

The images were taken as shown in figure 5.1, 400 images were captured at different light conditions including low light, high light, and at various distances ranging from 10 cm to 50 cm. The test set included plates with normal, dirt, and other noise distortion. The test was performed on all images, for each image a manual inspection was made, to see if the license plate region has been successfully extracted.

5.3.2 Results

The proposed method based on vertical edge matching turned out to be an efficient way of detecting the license plate region. As table 5.1 shows that, the algorithm was capable of extracting the plates at different distances, the best results appeared within the range of 25 cm to 35 cm away from the camera. At this distance the algorithm almost always found the plate. In only 8 of the 400 images, it was not capable of locating the plate region. This is because, the edges were not detected properly. This is due to the very harmed plate, and the very bright state, the algorithm at these conditions couldn't identify the edges and consequently locate the plate.

In the images that were taken under any other range, the results show a less rate of success. This is due to the bad quality of the acquired image, which made it difficult for the algorithm to detect the edges, and extract the plate.

In a conclusion, the suggested method will guarantee a very high success rate within the range of 25 cm to 35 cm, and according to these experimental results the location of the photocell will be placed within this range to sense the arrival of the vehicle, to give the best object detection and plate extraction results.

Method	10 cm-25 cm	25 cm-35 cm	35 cm -50 cm
Vertical edge matching	150/400	392/400	340/400

Table 5.1 Results of extraction phase

5.4 Segmentations test

This section describes the test performed for isolation of characters using horizontal and vertical projection. The purpose of this method is to divide the plate into seven regions, each region contains an individual character. A successful isolation must fulfill all of the following criteria:

- The plate must be divided into seven sub images.
- None of the seven characters have significant loss of details.
- The order of those seven images must appear in the same way they appear in the plate.

Figure 5.2 shows an example of successful and unsuccessful character isolation.

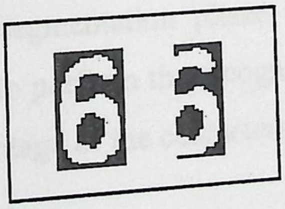


Figure 5.2 a successful and unsuccessful segmentation

5.4.1 Test data and test description

The test is performed on the images received from a successful extraction phase. The process therefore, was performed on 392 images. Also for each image a manual inspection was made, to see if seven regions are isolated and met the previous criterion.

5.4.2 Results

The proposed method based on horizontal and vertical projection turned out to be a very robust way in dividing the plate into seven sub images, In the 392 images the algorithm succeeded in isolating all of them into seven regions, this means that the success rate for this method is 100%. As table 5.2 shows.

Method	Successful	Rate of success
Horizontal and vertical projection	392/392	100%

Table 5.2 Segmentation results

5.5 Recognition test

The final step in recognizing the license plate is identifying the single characters extracted from the segmentation phase. Neural approach and nearest neighbor are used in this phase to perform the recognition of the characters. For this test to be a success a high percentage of the characters must be identified correctly.

5.5.1 Test data and test description

The test data originates from 392 test images extracted and isolated in the segmentation phase. The actual test data are the single characters. Character are identified using neural approach, and nearest neighbor by extracting the relevant features of the character. First the results of the features that were extracted during recognition phase are shown, and based on those selected features the recognition results will be shown.

5.5.2 Features extraction results

Through the features extraction step several features were selected, according to the experimental analysis, the results showed that the horizontal projections along with the edges of each character are the best selected features as they gave the highest recognition rate.

At first the selected features presented the horizontal and vertical projections of each character as shown in figure 5.3, these were chosen on the base that horizontal projections are distinct for many of the digits. The results were not satisfying, as they gave a bad recognition rate as seen in table 5.3, the reason behind this is that vertical projections for most characters are similar in structure with either a single wide peak, or a peak in the beginning and end of the digit, this has decreased the success rate for recognition.

Features were then chosen as a combination of horizontal projection, area, and black to white ratio of each character, the results were also not efficient as table 5.3 shows a bad recognition rate.

Finally the horizontal projections along with the edges of each character were tested and gave a high recognition rate as shown in table 5.3. these were chosen to be the representative features.

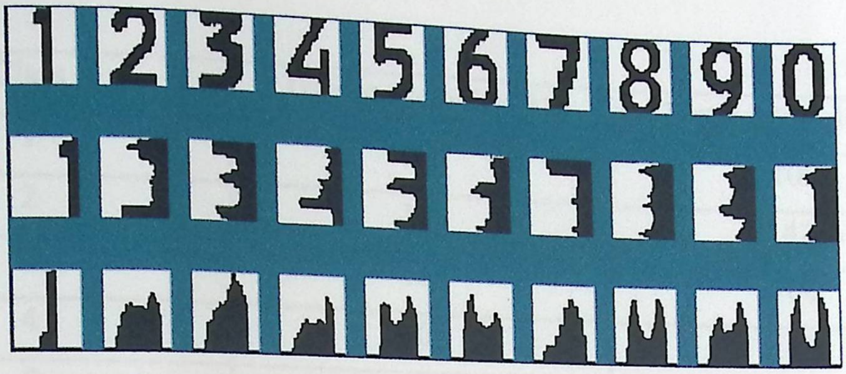


Figure 5.3 Horizontal and vertical projections

Features method	Horizontal , and vertical projections	Horizontal projections, area , and black to white ratio	Horizontal projections, and edges
Recognition rate % Using neural approach	85.423%	90.142%	95.844%
Recognition rate % Using nearest neighbor	80.123%	84.343%	87.636%

Table 5.3 Features extraction methods results

5.5.3 Neural Recognition results

The results of tests using neural approach was done using the two selected features, horizontal projections and the edges of each character. 105 different data sets for each of the 11 classes of the characters were created, 70 samples of them were used to train the neural network, and 35 were used for testing. Several trainings were made in order to choose the best results, and finally the best result gave a 95.844% rate of success. Table 5.4 shows the results of the final experiment for the recognition. To see all experiments with their results see appendix A.

Class	Training samples (%)	Testing Samples (%)
1	100	100
2	100	91.4286
3	100	100
4	100	100
5	98.5714	97.1429
6	100	100
7	100	100
8	95.7134	88.5714
9	98.5714	100
0	100	100
unknown	94.2857	77.1429
Average	98.8311	95.844

Table 5.4 Final experiment result using neural approach

5.5.4 Nearest Neighbor Results

Nearest neighbor algorithm was used in recognition as a classifier, this algorithm was tested on all the features that were selected at first, in each time the recognition rate was observed, to know the best given results. The final chosen features were horizontal projections, and edges. The results are shown in table 5.3. Looking on table 5.5, one can notice that the recognition rate is quietly good (87.6363%) but still couldn't be considered a reliable one since the recognition rate for character five is 65%, and the noise is 38%. These results will cause the system in many cases to fail in recognizing both characters which will affect the overall performance.

Class	Total samples	Recognition rate (%)
1	35	100
2	35	100
3	35	97
4	35	98
5	35	65
6	35	88
7	35	98
8	35	81
9	35	99
0	35	100
unknown	35	38
Average	35	87.6363

Table 5.5 Final experiment result using nearest neighbor

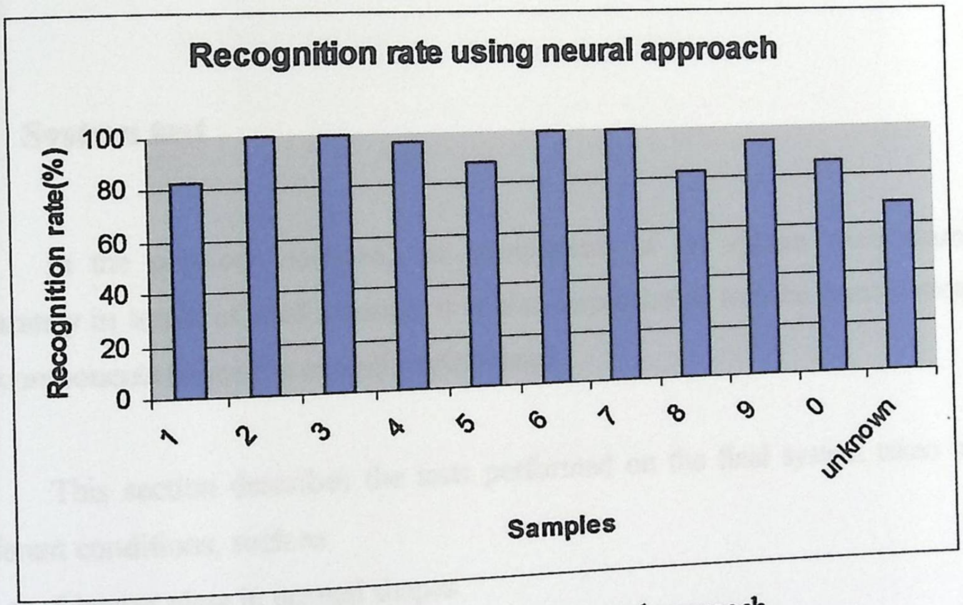
5.5.5 Comparison between neural approach and nearest neighbor approach

The two methods gave a good recognition results, but after the a lot of experimental results, and adding noise information to the original dataset to see how the system would behave at different conditions (normal, rain, dim light, night, scratches), the neural approach gave a more reliable recognition rate than the nearest neighbor.

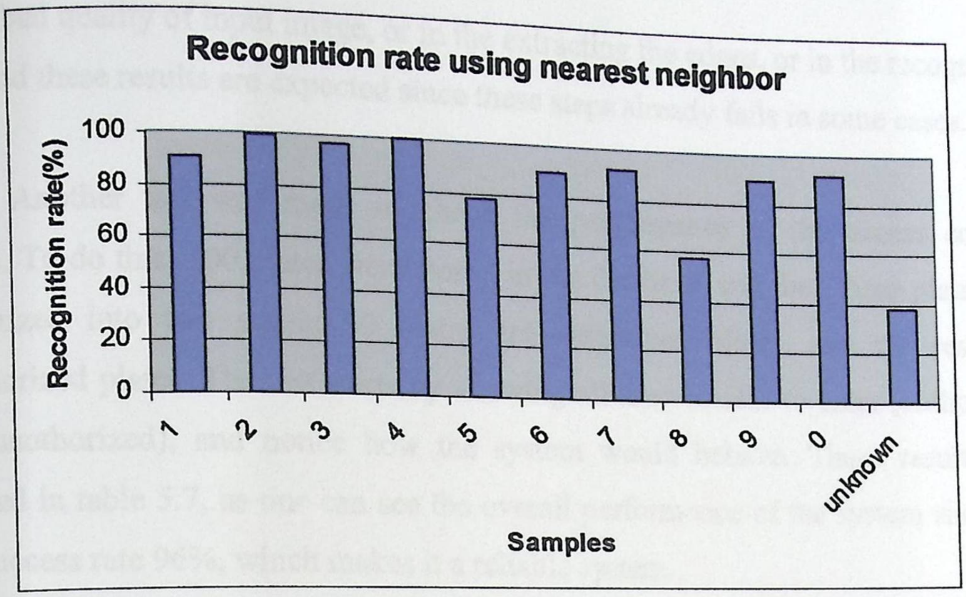
The nearest neighbor at these conditions resulted in bad recognition rate as shown in table 5.6 which makes this algorithm only reliable at normal, stable cases. This is a good reason to use the neural approach in character recognition. These results are shown in figure 5.4 (only for selected features).

Features method	Horizontal , and vertical projections	Horizontal projections, area , and black to white ratio	Horizontal projections, and edges
Recognition rate % Using neural approach	85.19%	86.12%	90.91%
Recognition rate % Using nearest neighbor	81.21%	82.12%	84.4%

Table 5.6 Recognition rate methods when noise information were added



(a) Recognition rate using neural approach



(b) Recognition rate using nearest neighbor

Figure 5.4 Recognition rate methods when noise information were added

5.6 System test

In the previous sections, the components of the system were examined separately in terms of performance. It is also important to test the combinations of the components to see the overall performance.

This section describes the tests performed on the final system taken under different conditions, such as

- License plate in normal shapes.
- Dirty license plate.
- License plates that have the same color as vehicle body.

The whole system is working good achieving a high recognition rate. The experimental results show that the shortcoming of the proposed system is mainly due to bad quality of input image, or in the extracting the edges, or in the recognition step, and these results are expected since these steps already fails in some cases.

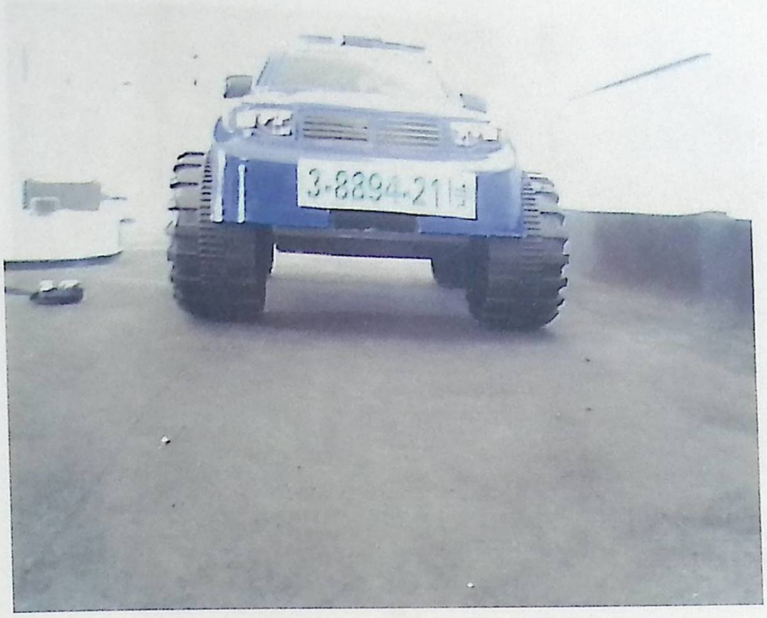
Another test was made to check the performance of the access control system. To do this, 100 plates were stored in the database, and then these plates are categorized into two states: 80 plates are authorized plates, and the rest are unauthorized plates. The test starts by allowing all the vehicles to enter (authorized and unauthorized), and notice how the system would behave. These results are recorded in table 5.7, as one can see the overall performance of the system shows a high success rate 96%, which makes it a reliable system.

Phases	Success rate%
Extraction Phase	98%
Segmentation	100%
Recognition	98%
Overall performance	96%

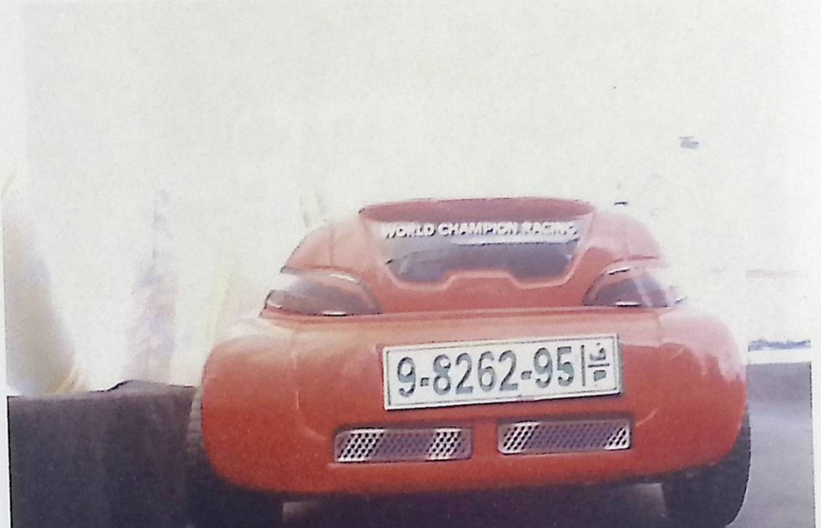
Table 5.7 Overall performance

5.7 Summary

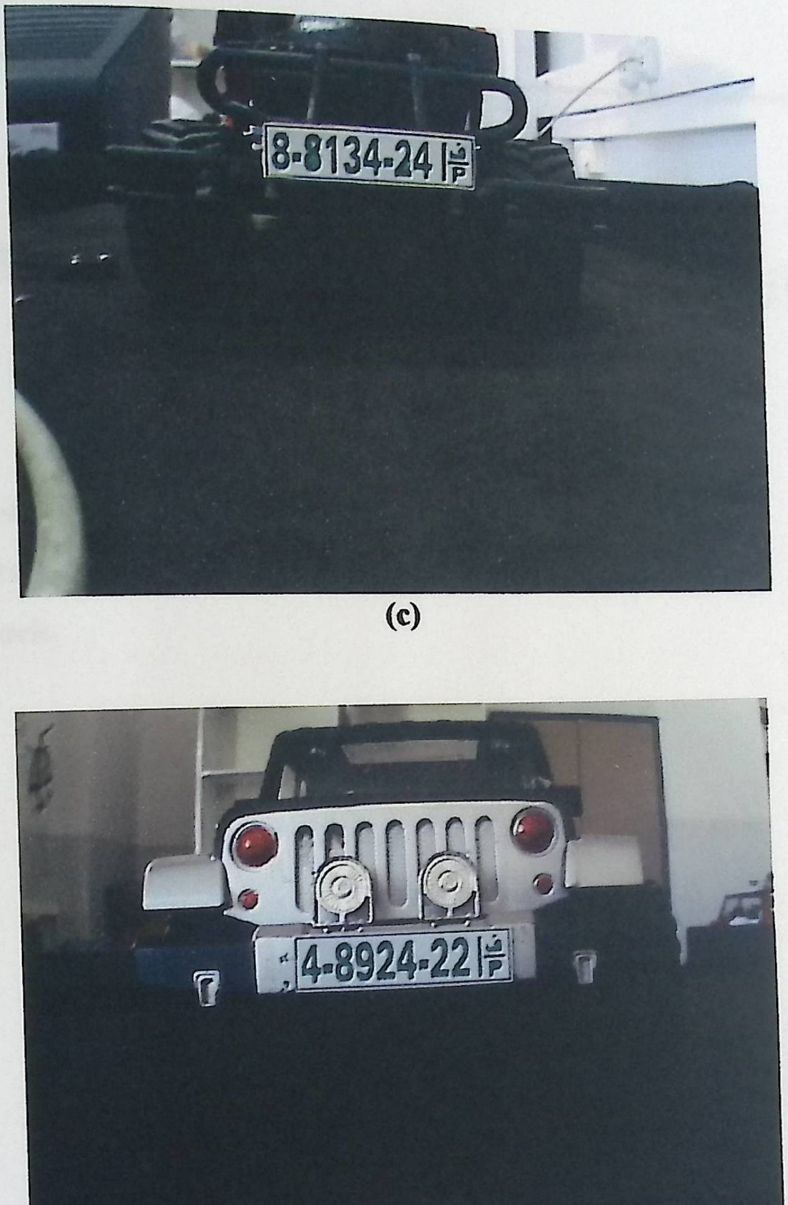
This chapter covered the tests on the individual phases of the LPR system. For the extraction method the proposed method showed good results. In the segmentation phase the horizontal and vertical projection turned out to be a robust way in dividing the plate into seven regions. And for the recognition the neural also gave good results. Then the overall performance of the system is tested, the results indicates that the methods used are efficient.



(a)



(b)



(c)

(d)

Figure 5.1 Images taken at different condition

CHAPTER SIX 6

Conclusions and Future work

6.1 Conclusion.

6.2 Challenges

6.3 Future work.

CHAPTER SIX

Conclusions and Future work

6.1 Conclusion

The main focus of this project is to experiment deeply the possibility of automating the whole process of license plate recognition and then implementing this system in controlling and allowing the access of the authorized vehicles that belong to a specific organization. Given an input image, the system extracts the license plate, isolates the characters, and finally identifies the characters. For each task, a set of methods were proposed, designed, and applied. For the extraction step the vertical edge matching is applied, and this method has proved its capability of extracting the plate from the image. In the segmentation step the horizontal and vertical projections are applied, this method turned out to be very successful. Finally, in the recognition step the neural approach is used and gave a high recognition rate. This means that the methods that were applied are good, and efficient as they gave satisfying results .The results obtained are shown in table 6.1.

Phases	Success rate
Extraction	98%
Segmentation	100%
Recognition	98 %
Overall performance	96 %

Table 6.1 Main results

6.2 Challenges

This section lists the problems were faced during the design process:

1. The main challenge we faced through our project was how to implement this system in reality. That demanded going frequently to the field to gather as many pictures as possible in order to set a database, which cost a considerable amount of money and was very much time consuming. To solve the problem a model was designed that simulates the original natural condition in terms of different changes that such a system could be exposed to.

2. Webcam connection through VC++, this was solved through using opencv library.

3. When images were taken at long distances, unclear images were acquired this has affected the outcome of the recognition phase. To solve this problem all images were captured within a certain range of distance away from camera position, which resulted in a higher recognition rate.

6.3 Future work

The following points are suggested as recommendations for future work:

- The system was tested on model not real images, so testing it on a real system is what recommended first to be done.
- The present system is designed to work on images taken from a certain distance, making the system distance free is a good extension.
- The camera is placed to capture images in only one direction (straightforward) making this system capable of capturing at different angels would make the system more reliable.

References

[1] Kim, G.M., (1997). "The Automatic Recognition of the Plate of Vehicle using Correlation Coefficient and Hough transform", Journal of Control, Automation and System Engineering, (vol.3, no.5, pp.511-519).

[2] Park, S,H., Kim, K.I., Jung, K., & Kim, H.J.(1999). "Locating Car license plate using Neural Networks", IEEE electronic letters, (vol.35,no17,pp.1475-1477).

[3] Duan, D., Hong, L, D., Vinch, P.C., & Nguyen, V.H.(2005)."Building an Automatic Vehicle License-Plate Recognition System", conference in computer science.

[4] Gonzalez, R, C., &Woods, R,E.(2004).Digital Image Processing. 2nd.ed, Pearson Prentice Hall, New Jersey.

[5] Morel, J.(1995). "Variational Methods in Image Segmentation", Birhauser, Boston.

[6] Hansen, H., Kristensen, A. W., Kohler, M.P., Mikklesen, A.W., Pedersen J. M., & Trangeled, M. (2002)." Automatic Recognition of License Plates", Institute for Electronic System, Aalborg University.

[7] Mei, Yu., Kim, Y.D.(2000). "An approach to Korean License Plate Recognition Based on Vertical Edge Matching", IEEE International Conference on Systems,(vol.4, pp.2975-2980).

[8] Natio, T. Tsukada, T. Yamada.(1999). " Robust recognition methods for inclined license plate under various illumination conditions outdoors", IEEE International Conference on intelligent transport systems, (pp. 697-702).

[9] Hu, M.K.(1962). " Visual Pattern Recognition by moment invariant", IRE Transaction on Information Theory, (pp.179-187).

[10] Mathews, J.(2004). Thresholding and segmentation. Retrieved October,11,2008. Website: http://www.generation5.org/content/2003/segmentation.asp.

[11] Gonzalez, R, C., Woods, R,E., & Eddins, S,L. (2004). Digital Image Processing Using Matlab. 2nd.ed, Pearson Prentice Hall, New Jersey.

[12] Electronic textbook stats soft.(2008).Neural networks. Retrieved October, 17,2008 Website: http://www.statsoft.com/textbook/stneunet.html#multilayer.

[13] El-Adawi, M., Keshk, H, M., &Haragi, M, M. (2004). "Automatic license plate recognition", Helwan University, Egypt.

[14] Wang, X. (2002). "Feature Extraction and Dimensionality Reduction in Pattern Recognition and their Application in Speed Recognition", Griffith University, Australia

[15] Electronic textbook stats soft.(2008).Neural networks. Retrieved April, 17,2009 Website: http://www.generation5.org/content/2002/im01.asp.

[16] The purple math forums.(2008). Slope of a straight line. Retrieved December, 17,2008 Website: http://www.purplemath.com/modules/slope.htm

[17] The purple math forums.(2008).The distance formula, Retrieved December, 17,2008 Website:http://www.purplemath.com/modules/distform.htm.

[18] The mathworks. (2009). Converting RGB image into a grayscale image, retrieved February, 22, 2009 Website http://www.mathworks.com/support/solutions/data/1-1ASCU.html.

Appendix A

This appendix reveals several tests that were performed using the two methods for the recognition phase. First the neural approach test results are shown in both cases (with and without noise), then the nearest neighbor approach test results are shown.

Neural approach test results
Selected features: Horizontal projections +edges
nput data = 70 (without noise)

arget error	1.00E-10	1.00E-05	1.00E-05
epochs	700	800	800
layer #	3	3	3
# neurons	70/28/11	70/28/11	70/28/11
func.	satlin / logsig	satlin / purelin	satlin / satlin
traning method	trainscg	trainscg	trainscg
traning samples #	70	70	70
testing samples #	35	35	35

	Training samples test	Test sample test	Training samples test	Test sample test	Training samples test	Test sample test
One	100	100	100	100	100	100
Two	100	100	100	85.7143	100	88.5714
Three	100	100	100	97.1429	100	97.1429
Four	98.5714	98.5714	100	88.5714	98.5714	88.5714
Five	98.5714	98.5714	98.5714	94.2857	98.5714	94.2857
Six	98.5714	98.5714	98.5714	91.4286	98.5714	82.8571
Seven	100	100	100	77.1429	100	82.8571
Eight	98.5714	98.5714	100	94.2857	98.5714	77.1429
Nine	98.5714	98.5714	100	100	100	100
Zero	100	100	100	94.2857	100	94.2857
Unknown	100	100	100	94.2857	100	82.8571
Average	99.42856	99.42856	99.74025455	92.69842	99.48050909	91.42857143

target error	1.00E-10	1.00E-05	1.00E-05
target error	1.00E-10	1.00E-05	1.00E-05
epochs	700	900	800
layer #	3	3	3
# neurons	70/28/11	70/28/11	70/28/11
func.	satlin / purelin	satlin / purelin	satlin / satlin
traning method	trainscg	trainscg	trainscg
traning samples #	70	70	70
testing samples #	35	35	35

	Training samples test	Test sample test	Training samples test	Test sample test	Training samples test	Test sample test
One	100	100	100	100	100	100
Two	100	91.4286	100	100	100	100
Three	100	100	100	100	100	100
Four	100	100	100	100	100	96.543
Five	98.5714	97.1429	98.5714	97.1429	98.5714	97.1429
Six	100	100	100	100	98.5714	100
Seven	100	100	100	100	100	100
Eight	95.7143	88.5714	95.7143	88.5714	100	89.5714
Nine	98.5714	100	98.5714	100	100	100
Zero	100	100	100	100	100	100
Unknown	94.2857	77.1429	94.2857	71.4286	100	77.1429
Average	98.8311636	95.84416364	98.8311636	96.1039	99.74025455	96.40002

Neural approach test results
Selected features: Horizontal projections +edges
Input data = 70 (with noise)

target error	1.00E-10	1.00E-10	1.00E-05
epochs	700	800	800
layer #	3	3	3
# neurons	70/28/11	70/28/11	70/28/11
func.	satlin / purelin	satlin / purelin	satlin / satlin
traning method	trainscg	trainscg	trainscg
traning samples #	70	70	70
testing samples #	35	35	35

	Training samples test	**Test sample test**	**Training samples test**	**Test sample test**	**Training samples test**	**Test sample test**
One	82.8571	82.8571	100	82.8571	82.8571	82.8571
Two	100	97.1429	85.7143	100	100	97.1429
Three	100	100	94.57	100	100	100
Four	94.2857	91.4286	85.7143	97.1429	94.2857	91.4286
Five	91.4286	88.5714	97.1429	88.5714	91.4286	88.5714
Six	94.2857	97.1429	88.5	100	94.2857	97.1429
Seven	94.2857	94.2857	77.1429	100	94.2857	94.2857
Eight	85.7143	82.8571	77.142	82.8571	85.7143	82.8571
Nine	94.2857	91.4286	97.129	94.2857	94.2857	91.4286
Zero	82.8571	77.1429	94.2857	85.7143	82.8571	77.1429
Unknown	62.8571	65.7143	97.1429	68.5714	62.8571	65.7143
Average	89.3506364	88.05195	92.89085	90.90908	89.3506364	88.05195

Nearest neighbor approach test results
Selected features: Horizontal projections +edges
Input data = 70 (without noise)

	Test sample test	Test sample test	Test sample test
One	100	86	100
Two	99	100	100
Three	99	99	97
Four	100	100	98
Five	63	65	65
Six	88	88	88
Seven	98	98	98
Eight	74	81	81
Nine	99	89	99
Zero	100	95	100
Unknown	40	38	38
Average	87.2727273	85.36364	87.63636

Nearest neighbor approach test results
Selected features: Horizontal projections +edges
Input data = 70 (with noise)

	Test sample test	Test sample test	Test sample test
One	87	87	89.987
Two	90	87	90
Three	99	99	99
Four	100	89.543	89
Five	65	65	65
Six	88.98	88.98	88
Seven	97	94	98
Eight	79.5	79	76.546
Nine	89	89	87.654
Zero	95.2	95	76.341
Unknown	38	55	38
Average	84.42545	84.41118	81.59345