University of the Philippines Manila
College of Arts and Sciences
Department of Physical Sciences and Mathematics

PLATE NUMBER RECOGNITION SYSTEM FOR BEATING THE RED LIGHT VIOLATORS

A Special Problem in partial fulfillment
of the requirements for the degree of
Bachelor of Science in Computer Science

Submitted by:

Rikki Ruperto N. Robles
April 2010

ACCEPTANCE SHEET

The Special Problem entitled "Plate Number Recognition System for Beating the Red Light Violators" prepared and submitted by Rikki Ruperto N. Robles in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science has been examined and is recommended for acceptance.

_____
Gregorio B. Baes, Ph.D.
Adviser

(candidate).

EXAMINERS:

	Approved	**Disapproved**
1. Avegail D. Carpio, M.S.	_____	_____
2. Richard Bryann L. Chua, M.S.	_____	_____
3. Aldrich Colin K. Co, M.S. (candidate)	_____	_____
4. Ma. Sheila A. Magboo, M.S	_____	_____
5. Vincent Peter C. Magboo, M.D., M.S.	_____	_____
6. Geoffrey A. Solano, M.S.	_____	_____
7. Bernie B. Terrado, M.S. (candidate)	_____	_____

Accepted and approved as partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science.

_____
Geoffrey A. Solano, M.S.
Unit Head
Mathematical and Computing Sciences Unit
Department of Physical Sciences and Mathematics

_____
Marcelina B. Lirazan, Ph.D.
Chair
Department of Physical Sciences and Mathematics

_____
Reynaldo H. Imperial, Ph.D.
Dean

College of Arts and Sciences

ABSTRACT

A plate number recognition system is a computer based system, integrated with the red light system, that is designed for the identification of the plate number of a vehicle violating the *beating the red light* rule.

The system has been developed to augment the current manual apprehension system of the Metro Manila Development Authority (MMDA) in apprehending erring motorists. The system addresses issues on potential bribery between motorists and traffic enforcers and situation where in no traffic enforcers could monitor a particular intersection.

It possesses functionalities that are under the fields of computer vision and neural networks. This includes Edge Detection, Connected Components, Morphological processes, Feed-forward neural network and the Backpropagation method of training. The edge detection is used in the process of localizing plate number candidates which is helped by the different morphological methods. Connected Components segment the localized plate number candidates to separate the potential characters of a plate number. To recognize the segmented characters, they are then used as inputs for the feed-forward neural network trained by the backpropagation method.

Keywords: Plate number recognition system, red light system, beating the red light, computer vision, neural networks, edge detection, connected components, feedforward, backpropagation

TABLE OF CONTENTS

I. Introduction

1.1 Background of the Study

The Metro Manila Development Authority (MMDA) is an agency in the Philippines which performs planning, monitoring and coordinative functions, and in the process exercise regulatory and supervisory authority over the delivery of metro-wide services within Metro Manila without reducing the autonomy of the local government units concerning purely local matters. Some of these services include: (1) Development planning, (2) Transportation and traffic management, (3) Solid waste disposal and management, and (4) Flood control and sewerage management just to name a few. [1]

In line with one of these services, specifically traffic management, the agency has the authority to implement all traffic enforcement operations and policies. [1] If an MMDA traffic enforcer witnesses a traffic violation, he/she can issue violation receipts or tickets to the erring motorist.

'Beating the red light' is an example of traffic infraction that the agency apprehends. This violation is incurred when a motorist runs through an intersection if the traffic light has already turned red. The violation may fall under reckless driving or disregard of traffic signs – depending on the speed of the vehicle.

There are a lot of factors as to why motorists would run a red light. It might be that these motorists are trying not to be late for office or an appointment, or they're catching a flight, or

there are no cars and they think they're safe to pass especially during wee hours of the morning. These people, of course, know that this is wrong, but why do they still do it? Mainly because there is no traffic enforcers present to apprehend them.

Maybe these motorists are lucky enough to evade the law one time or another, but repercussions are inevitable. Accidents abound because of traffic violations like 'beating the red light'. These accidents range from a minor collision with another vehicle to death of pedestrians, passengers or the drivers themselves.

In the likely event, though, that a traffic enforcer is present and a motorist commits the infraction, the violator has no choice but to pull over and be issued a violation receipt by the enforcer. But then there is the concept of *'kotong'*, wherein the enforcer would ask for money instead of issuing a ticket for the violator. This current system of the MMDA for apprehending 'beating the red light' violators greatly contributes to corruption, perpetuates bribery and cannot apprehend violators for situations when traffic enforcers are absent.

So what this paper provides is an enhancement to the current system of the agency from manual apprehension to computerized identification of violators. The developed system is a plate number recognition system for 'beating the red light' violators.

1.2 Statement of the Problem

General

The manual apprehension system of MMDA does not provide evidence in the occurrence of the event, relies only on the eye of the enforcer and only has thin deployment of enforcers.

Specific:

The manual apprehension system of MMDA for 'beating the red light' is not very flexible. It does not provide evidence of the occurrence of the event making it hard to prove that such a violation happened. Violators can easily get away from this. In this system also, the traffic enforcers only use their naked eye which could be subject to high percentage of error like misreading of the plate number of the vehicle. This can be due to factors like the vision of the enforcer, over-speeding of the car which makes it hard for the enforcer to read the plate number or the quality of the plate number. The system also only provides thin deployment of traffic enforcers. The coverage is only limited to certain areas

1.3 Objectives of the Study

1. Localize the plate number region of the car through image processing.
    a. Use Second Derivative Approximating function to find the edges of the image.
    b. Use Running-sum algorithm to find plate number candidates by determining the area of highest edge concentration.
2. Segment the characters from the localized plate number region
    a. Use Connected Components labelling to group the alpha numeric characters within the plate number region

b. Create a 25x15 image for each character.

3. Recognize the segmented characters using Neural Networks

a. Use Backpropagation algorithm for training the neural networks

b. Use feed-forward neural network in recognizing the 25x15 images as input patterns

1.4 Significance of the Study

Since the current apprehension system of the MMDA for 'beating the red light' violators is all manual, the issues of corruption and unavailability of traffic enforcers at certain intersections come about. Examples of when corruption happens are: when a traffic enforcer compels a violating motorist into giving money instead of issuance of violation receipt, and when the enforcer uses the more subtle approach of dropping subtle remark indicating his/her preference of receiving money. So with the reduced number of traffic enforcers needed at every intersection mainly due to the presence of the proposed system integrated with red-light cameras, opportunities for corruption are lessened.

The issue of unavailability of traffic enforcers in manual apprehension system is also a great opportunity to show how the system is of great help in maintaining traffic management and policies at road intersections. With the developed system, even in the absence of a traffic enforcer, management of traffic and apprehension of erring motorists are still in operation.

Although, in employing this system, the need for traffic enforcers is greatly lessened, they are still expected to be monitoring at some areas – the reason practically being so that in cases of manual overrides of the traffic situation or emergencies, they are quick to respond.

1.5 Scope and Limitation

Listed below are the scope and limitations of the system:

1. The developed system is integrated with red-light cameras that captures images of 'beating the red light violators'. These red-light cameras are attached to traffic lights and are triggered to capture images at intersections when the traffic light is lit red and a motorist runs through it.

2. The developed system only receives images and outputs the plate number recognized. Any further actions by the MMDA with the data output or follow-up of the penalties are not the concern of the system.

3. Only the plate numbers regularly issued by the LTO for cars, vans and trucks is processed by the proposed system. Commemorative, diplomatic and government plates are not included.

4. Vehicles without plates or for registration are outside the scope of the system.

5. Cases like when two vehicles are caught by the camera where one of them committed the violation while the other came from the right direction (the road where the traffic light is green) are not within the scope of the proposed system.

6. The case of when a traffic enforcer is compelled to manually override the flow of traffic at an intersection and this act disrupts the technicalities of 'beating the red light' is not also within the scope of the proposed system.

7. If a plate number is dilapidated or obstructed to the point that it's virtually improbable to recognize, then the proposed system is not expected to perform well.

8. Performance of the system is dependent on the quality (including the lighting) of the image caught by the camera.

II. Review of Related Literature

The Plate Number Recognition System is an application to be used for identifying beating the red light violators in Metro Manila. It takes in images of cars passing by the main streets of Manila as inputs to be processed. This is done by analyzing the video, frame by frame, and retrieve the frame where possible violation occurred.

Apart from this paper's proposed system, there have been others that has discussed and ventured on making their own Plate Number Recognition System.

One example would be a Malaysian Vehicle License Plate Localization and Recognition System [2]. The system is developed based on digital images and is applied to commercial car park systems for the use of documenting access of parking services, secure usage of parking

houses and also to prevent car theft issues. The localization algorithm is a mixture of morphological processes while the recognition is attained by implementing the feed-forward backpropagation artificial neural network.

Another system is an Automatic Vehicle Identification (AVI) by Plate Recognition developed by Serkan Ozbay and Ergun Ercelebi [3]. Their system's algorithm consists of 3 parts: extraction of plate region, segmentation of characters and recognition of the plate characters. Edge-detection algorithm, smearing algorithm and statistical based template matching are what the system uses for the major parts respectively.

License Plate Number Recognition proposed by Garcia-Osorio, Diez-Pastor, Rodriguez and Maudes is a system designed to work in not so restricted and structured environment [4]. The application can identify plate numbers independently of its size, orientation, position or lightening condition.

A Novel Fuzzy Multilayer Neural Network is used in a different License Plate Recognition application [5]. This system undergoes three stages in recognizing license plates.

First Stage: plate is detected inside the digital image

Second Stage: characters are extracted by means of horizontal and vertical projections

Third Stage: fuzzy neural network is used to recognized the license plates

The system proves to be robust as compared to other license plate recognition systems.

An application used for recognizing Persian license plates is called a Farsi License Plate Recognition system [6]. This application undergoes the same stages as the other systems of the same functions. This system has been tested on 400 vehicle images with the rate of success recognition of 95%.

Another localised number plate recognition system is one developed for Indonesian vehicles [7]. It is developed to increase the efficiency of several traffic related services like automated parking, traffic light surveillance, electronic toll collection, and vehicle surveillance for police uses. Unfortunately, the system does not perform up to its expectations. It still requires a lot of human supervision and input, and it depends heavily on the training set used.

One paper proposes a robust system to recognize plate numbers by multi-frames learning [8]. It uses a morphology-based method to extract contrast features to find possible candidates for license plates. After locating the region of the license plate, the scheme of shape contexts is used to recognize the characters in the plate. Basically, it is a Recognition of Vehicle License Plate from a Video Sequence.

The use of license plate recognition systems in e-Government has been proven helpful [9]. And this is what Wu, et al, have confirmed in their research. They have also dealt with the issue on how to apply license plate recognition in e-Government in order to improve performance.

Another distinct way of recognizing license plates is by using decision trees. Janota, et al, implemented this way in their proposed system [10]. The system also focuses on processing only one captured vehicle. The algorithm used in this system finds holes and arcs to recognize characters. The system has also been prototyped in C++ language.

One last related application to this paper's proposed system is the one developed by Vazquez, et al [11]. They also use multilayer neural network to identify the symbols of the number plate, just like the previous applications. It also consists of two processes: the training and recognition processes. The training process is the stage where the application learns to

identify symbols from number plates. The recognition process, on the other hand, is the stage where the application actually identifies the characters in number plates.

III. Conceptual Framework

1. Plate Number Recognition System

Automatic Vehicle Identification (AVI) has many applications in traffic systems (highway electronic toll collection, red light violation enforcement, border and customs checkpoint, etc.). Plate number recognition is an effective form of AVI systems.

Generally, a plate number recognition system is made up of three (3) modules; license plate localization, character segmentation and character recognition.

1. License plate localization

The first step in a process of automatic number plate recognition is a detection of a plate number area. Humans define a plate number in a natural language as a "small plastic or metal plate attached to a vehicle for official identification purposes," but machines do not understand this definition as well as they do not understand "vehicle", "road", or whatever else is. Because of this, there is a need to find an alternative definition of a number plate based on descriptors that will be comprehensible for machines. [12]

a. Define the plate number as "rectangular area with increased occurrence of edges." The high density of edges on a small area is, most of the time, caused by the contrast of the colours of alphanumeric data in a plate number and the plate itself.

b. This process can sometimes detect a wrong area (not a plate number) that's why detection of several candidate regions is done.

c. The best among the candidates will be chosen by further heuristic analysis.

2. Character segmentation

The next step after the detection of the plate number area is the segmentation of the plate.

a. Plate number area is segmented into its constituent parts obtaining the characters individually.

b. Determine which pixels are the characters within the plate number area by using connected components labelling.

3. Character Recognition

The final step in the plate number recognition system. This module utilizes one of the most common mathematical models in Optical Character Recognition, the artificial neural network.

2. Computer Vision

Computer Vision is the enterprise of automating and integrating a wide range of processes and representations used for visual perception. It includes as parts many techniques that are useful themselves, such as image processing, statistical pattern classification, geometric modelling and cognitive processing. [19]

As a scientific discipline, computer vision is concerned with the theory of building artificial systems that obtain information from images. The image data can take many forms, such as video sequences, views from multiple cameras, or multi-dimensional data from a medical scanner.

3. Red Light Camera

Red light cameras are devices which are designed to take snapshot of a vehicle that illegally goes through an intersection where the light is red. These cameras help to enforce traffic laws by automatically photographing vehicles disobeying stop lights. The system continuously monitors the traffic signal and the camera is triggered by any vehicle entering the intersection above a preset minimum speed and following a specified time after the signal has turned red. There are 5,000 – 6,000 photo enforced red light cameras and speed cameras operating in the U.S. [20]

The system usually includes three (3) essential elements:

1. One or more cameras
2. One or more triggers
3. A computer

The computer is the brain behind the operation. It is wired to the cameras, the triggers and the traffic light circuit itself. The computer constantly monitors the traffic signal and the triggers. If a car sets off a trigger when the light is red, the computer tells the camera to take picture of the intersection.

This is how the red-light camera works. [21]

4. Artificial Neural Network

An Artificial Neural Network (ANN) is an information processing paradigm that is inspired by the way biological nervous systems, such as the brain, process information. The key element of this paradigm is the novel structure of the information processing system. It is composed of a large number of highly interconnected processing elements (neurons) working in unison to solve specific problems. ANNs, like people, learn by example. An ANN is configured for a specific application, such as pattern recognition or data classification, through a learning process. Learning in biological systems involves adjustments to the synaptic connections that exist between the neurons. This is true of ANNs as well. [14]

4.1 Architecture of Artificial Neural Network

An artificial neural network consists of a number of very simple processors called neurons, which are analogous to the biological neurons in the brain. These artificial neurons are connected by weighted links passing signals from neuron to another.

The output signal is transmitted through the neuron's outgoing connection. The outgoing connection branches out and transmits the same signal. The outgoing branches terminate at the incoming connections of other neurons in the network.

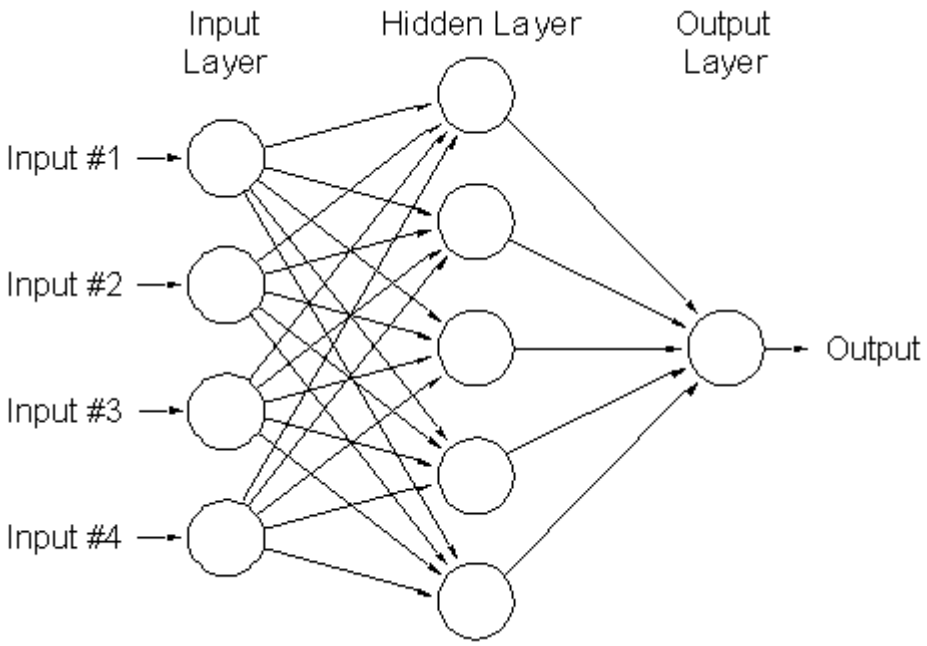

Figure 1. Architecture of an Artificial Neural Network

4.2 Feed-forward neural networks

Feed-forward neural networks consist of an input layer of source neurons, at least one middle or hidden layer of computational neurons, and an output layer of computational neurons. The input signals are propagated in a forward direction on a layer-by-layer basis; that is from input to output.

They are extensively used in pattern recognition. This type of organization is also referred to as bottom-up or top-down. [14]

4.3 Backpropagation Algorithm

Backpropagation algorithm is a common method of teaching artificial neural networks how to perform a given task. The algorithm has two (2) phases.

First, a training input pattern is presented to the network input layer. The network propagates the pattern from layer to layer until the output pattern is generated by the output layer. Second, if this pattern is different from the wanted output, an error is calculated and propagated backwards from the output layer to the input layer. The weights are modified as error is propagated. These are repeated until the stopping criterion is satisfied.

5. Digital Image Processing

Image processing is the analysis, interpretation and manipulation of signals in the form of images or with images as the input.

Digital image processing on the other hand is using computer algorithms to do image processing on digital images (usually stored as two dimensional arrays of pixels or pixel matrix).

5.1 Edge Detection

Edge detection refers to the process of identifying and locating sharp discontinuities in an image. The discontinuities are sudden changes in pixel intensity which characterize boundaries of objects in a scene.

5. 2 Connected Component Labelling

Connected components labelling scans an image and groups its pixels into components based on pixel connectivity, i.e. all pixels in a connected component share similar pixel intensity values and are adjacent to each other. [17]

With this method, the alpha-numeric characters of the localised plate number will be grouped into components and separated from each other. Images of the segmented characters with equal dimensions will be used for optical character recognition.

IV. Design and Implementation

To achieve the goals of the system, different algorithms are used in each module. For the plate number localization module, the second derivative approximating algorithm is used for the edge detection and the *running-sum* algorithm in finding the area with highest sum of edges. The character segmentation module uses the *connected components labelling* algorithm to segment the alpha-numeric characters of the localized plate number. And finally, the *feed-forward neural network* is used for the character recognition module trained by the *backpropagation algorithm*.

1. Plate Number Localization

1.1 Edge Detection using second derivative approximating function: [18]

1.1.1 Given input JPEG image *I[m][n]*, where m is the number of rows and n is the number of columns.

1.1.2 Convert *I* to grayscale image *GR[m][n]*

a) For $i = 0$ to $m - 1$

b) For $j = 0$ to $n - 1$

c) $GR[i][j] = (Red(I[i][j])+Green(I[i][j])+Blue(I[i][j]))/3$

0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	60	65	60	0
0	0	65	0	0
0	0	60	0	0
0	0	0	0	0

Figure 2. Grayscale GR[8][5]

1.1.3 Using *GR[m][n]* find the second derivative approximation *G[m][n]*

a) For $i = 0$ to $m - 1$

b) For $j = 0$ to $n - 1$

c) $G[i][j] = abs(GR[i-2][j] - 2(GR[i][j]) + G[i+2][j])$

0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
65	60	130	60	65
65	0	130	0	65
60	0	120	0	60
0	0	0	0	0

Figure 3. G[i][j]

1.2 Running-Sum Algorithm in finding the area with highest sum of edge [18]

1.2.1 Given intensity gradient matrix *G[m][n]* and bounding area *W x H*, create an array *V[m][n-H+1]*

1.2.2 Each cell value in *V[i][j]* is the sum of values in *G[i][j]* to *G[i+k][j]*

a) For *j = 0* to n-*H+1*

b) For *i = 0* to *m*

c) For *k = 0* to *H - 1*

d) *V[i][j] += G[i+k][j]*

0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
65	60	130	60	65
130	60	260	60	130
125	0	250	0	125
60	0	120	0	60

Figure 4. V[7][5]

1.2.3 Create another array *A[m-W+1][n-H+1]* where each cell value is the sum of values in *V[i][j]* to *V[i][j+k]*

a) For *j* = 0 to *n-H+1*

b) For *i = 0* to *m-W+1*

c) For *k = 0* to *W-1*

d) *A[i]j] += G[i][j+k]*

0	0	0

0	0	0
0	0	0
255	250	255
450	380	450
375	250	375
180	120	180

Figure 5. A[7][3]

1.2.4 Find the cell in *A* that has the greatest value, this is the area of highest edge concentration. The indices of the cell would be the corresponding indices of the north east point of the bounding area.

1.2.5 For multiple plate candidates, once a maximum area is located, the intensity values within its area are set to 0 and the running-sum algorithm is repeated.

1.2.6 Adjust the bounding area so that the center would be the row of the characters of the plate number. This is done by finding the center of the character row which is believed to be the center of mass of the edge values or simply the point where the weights are concentrated or balanced. The new localized area will be *L[H][W]*.

$$center_X = \frac{\sum_{x=0}^{W-1}\sum_{y=0}^{H-1} xE(x,y)}{\sum_{x=0}^{W-1}\sum_{y=0}^{H-1} E(x,y)}$$

$$center_Y = \frac{\sum_{x=0}^{W-1}\sum_{y=0}^{H-1} yE(x,y)}{\sum_{x=0}^{W-1}\sum_{y=0}^{H-1} E(x,y)}$$

Figure 6. Formula for finding the center of mass

1.2.7 Resize the bounding box so that it would include the alpha-numeric data within that region. The updated localize area will be *updatedL[H][W]*

60	65	60
0	65	0
0	60	0

Figure 7. updatedL[3][3]

2. Character Segmentation

2.1 Image Binarization

2.1.1 Given a predefined threshold θ, create binary image *B[H][W]*

a) For $i = 0$ to $H-1$

b) For $j = 0$ to $W-1$

c) If ($L[i][j] < \theta$) /* *if intensity is less than threshold* */

d) *B[i][j] = 0 /* white */*

e) Else

f) *B[i][j] = 1 /* black */*

1	1	1
0	1	0
0	1	0

Figure 8. B[3][3]

2.2 Connected Components Labelling [17]

2.2.1 Given the binary image *B[H][W],* look for the connected components

a) For *i = 0* to *H-1*

b) For *j = 0* to *W-1*

c) If (*B[i][j]* == 1)

d) If neighbouring pixels value = 1

e) Get label of neighbour

f) Else

g) Assign unique label for *B[i][j]*

Figure 9. Labelled Component

2.2.2 After finding the connected components, segment each component with equal dimensions.

3. Character Recognition

3.1 Backpropagation algorithm for training the neural network

3.1.1 Given set of training patterns (from A-Z for the neural network for letters, 0-9 for the neural network for digits) sized 25x15, randomized values for weights and thresholds distributed inside a small range, and a learning rate of 0.1, train the network to learn the alpha-numeric patterns

3.1.2 Initialize the input neurons *Input[375]*

a) /* Neural Network consists of 375 neurons for the input layer, 53 neurons on its hidden layer and 26 neurons for the output layer of the neural network for letters and 10 for the digits */

b) For *counter = 0* to *374*

c) For *i = 0* to *24*

d) For *j = 0* to *14*

e) *Input[counter] = Pattern[i][j]*

3.1.3 Propagate the input pattern until the output pattern is generated by the output layer

a) /* Input Layer to Hidden Layer */

b) For *i = 0* to *203*

c) For *j = 0* to *374*

d) *Temp1 += Input[j] * weightInput2Hidden[j][i]*

e) *Temp2 = Temp1 – ThresholdHidden[i]*

f) *Hidden[i] = ((2*1.716)/1+exp(-1 * 0.667 * Temp2)) – 1.716*

g) *Temp1, Temp2 = 0;*

h) /* Hidden Layer to Output Layer */

i) For *i = 0* to 35

j) For *j = 0* to *203*

k) *Temp1 += Hidden[j] * weightHidden2Output[j][i]*

l) *Temp2 = Temp1 – ThresholdOutput[i]*

m) *Output[i] = ((2*1.716)/1+exp(-1 * 0.667 * Temp2)) – 1.716*

n) *Temp1, Temp2 = 0*;

3.1.4 Calculate error $E_{output}[35]$ and propagate backwards through the network from output layer to the input layer. Modify the weights and threshold as error is propagated.

a) /* Calculate error E_{output} */

b) For $i = 0$ to 35

c) $E_{output}[i] = Target[i] - Output[i]$

d) /* Calculate error gradient for output layer δ_{output} */

e) For $i = 0$ to 35

f) $\delta_{output}[i] = Output[i] * [1 - Output[i]] * E_{output}[i]$

g) /* Calculate weight corrections */

h) For $i = 0$ to 35

i) For $j = 0$ to 203

j) $\Delta w[j][i] = 0.95 * \Delta w[j][i] + 0.1 * Hidden[j] * \delta_{output}[i]$

k) /* Update weights at output neurons */

l) For $i = 0$ to 35

m) For $j = 0$ to 203

n) $weightHidden2Output[j][i] = weightHidden2Output[j][i] + \Delta w[j][i]$

o) /* Calculate error gradient for neurons in hidden layer */

p) For $i = 0$ to 203

q) For $j = 0$ to 35

r) $E_{hidden} += (\delta_{output}[j] * weightHidden2Output[i][j])$

s) $\delta_{hidden}[i] = Hidden[i] * [1 - Hidden[i]] * E_{hidden}$

t) /* Calculate weight corrections */

u) For $i = 0$ to 203

v) For $j = 0$ to 374

w) $\Delta w[j][i] = 0.95 * \Delta w[j][i] + 0.1 * Input[j] * \delta_{hidden}[i]$

x) /* Update weights at the hidden heurons */

y) For $i = 0$ to 203

z) For $j = 0$ to 374

aa) $weightInput2Hidden[j][i] = weightInput2Hidden[j][i] + \Delta w[j][i]$

3.1.5 Do this until sum of squared errors < 0.001

3.2 Proceed with next pattern for training. **Feed-forward neural network**

3.2.1 Given segmented components from *B[H][W]*, use these as input patterns for the feed-forward neural network to recognize.

3.2.2 Run the neural network and get the output.

3.2.3 The node from the output layer that has the maximum value constitutes the character of the input pattern.

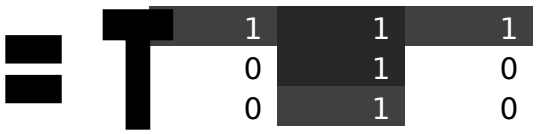

Figure 10. Input pattern = T

4. Technical Architecture

4.1 The proposed project is a stand-alone system running in any Windows platform.

4.2 The algorithms are implemented using the C++ programming language.

4.3 The source code is compiled using the Cygwin – a Linux-like environment for Windows.

4.4 GUI created using Windows 32 API.

V. Results

The Plate Number Recognition System initial screen basically provides a textbox where the path of the image to be recognized is entered. This is solely for presentation purposes as the program is expected to have its images entered automatically. It also has a menu bar at the top where one can find the author of the program and the option to end the program.

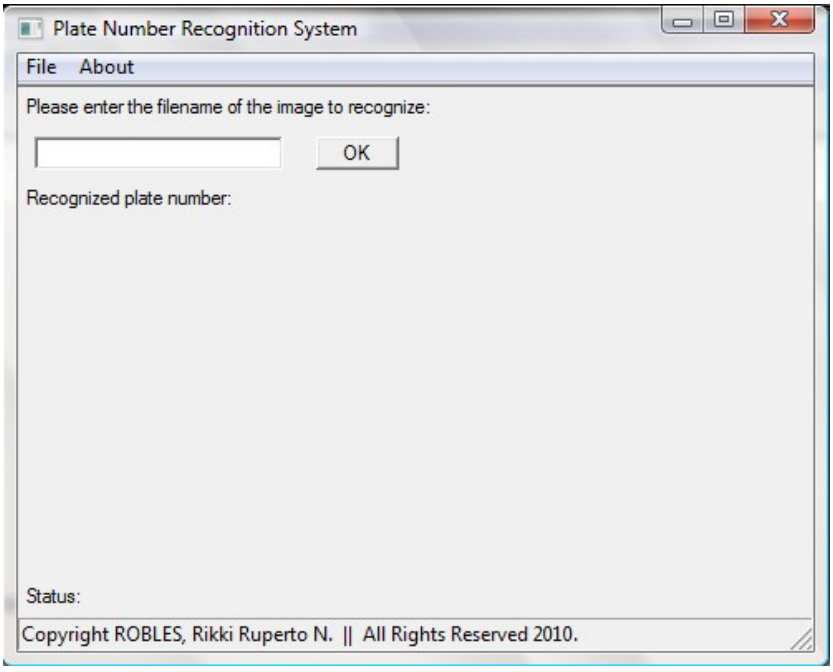

Figure 11. Initial Screen

A user could enter the specific path of the image to be recognized on the text box. Shown as such:

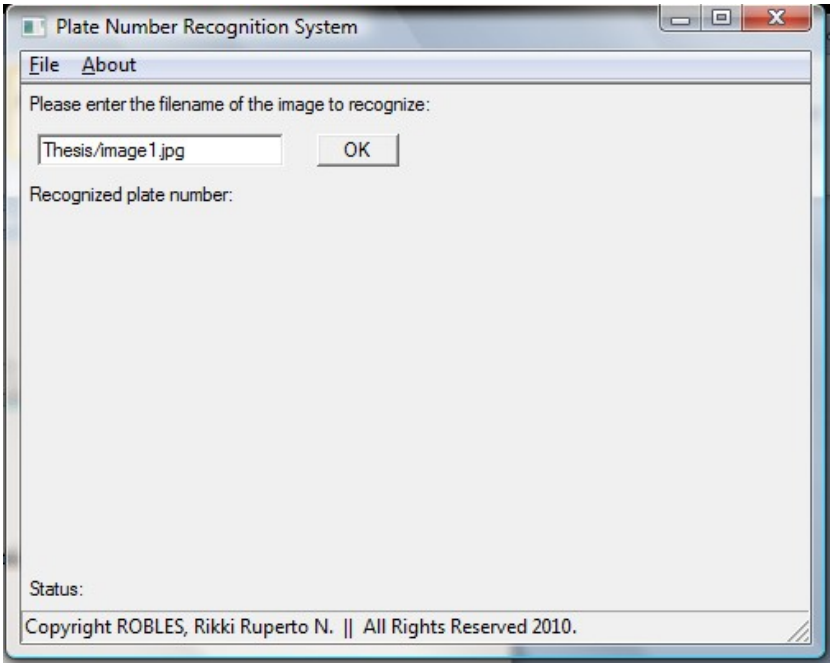

Figure 12. Screen with filepath

Pressing the OK button starts the plate number recognition process. It begins with edge detection method. The status of the function is shown at the bottom of the interface.

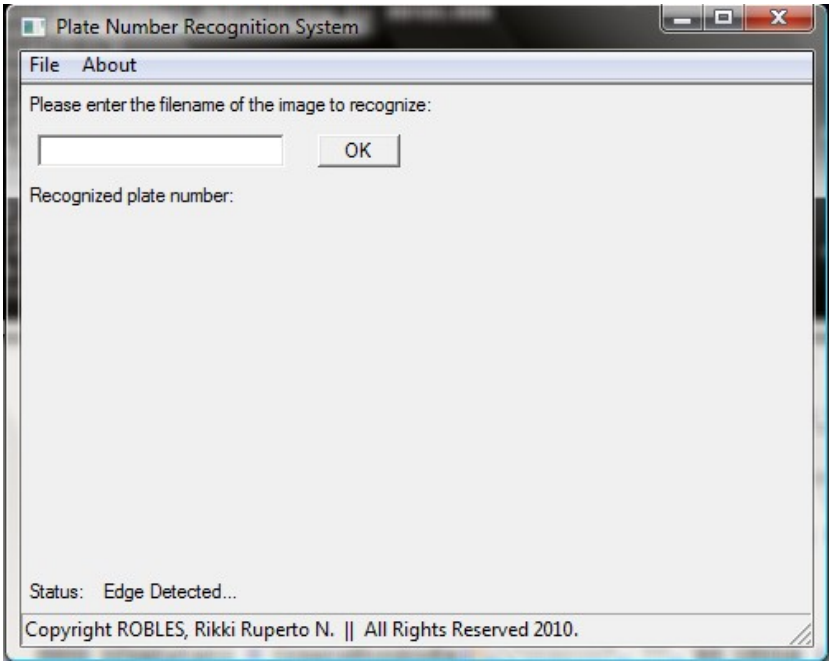

Figure 13. Recognition process on-going

The image being processed by the program is first converted into grayscale. After which, the edges of the grayscaled image is acquired then binarized.

Figure 14. Edge Detected

Figure 15. Binarized Edges

Ten plate candidates are then chosen from the binarized edge detected image. This is done by employing the running sum algorithm on the binarized image. The localized plate candidates are enclosed by the rectangles as shown:

Figure 16. Bounded Plate Candidates

Each plate candidate is processed for character recognition.

Figure 17. Plate Candidates

The methods that the grayscaled versions of these plate candidates undergo before the character recognition phase are as follows: improvement of contrast, binarization, and connected components acquisition.

Improve contrast:

Figure 18. Improved Contrast of Plate Candidates

Binarization:

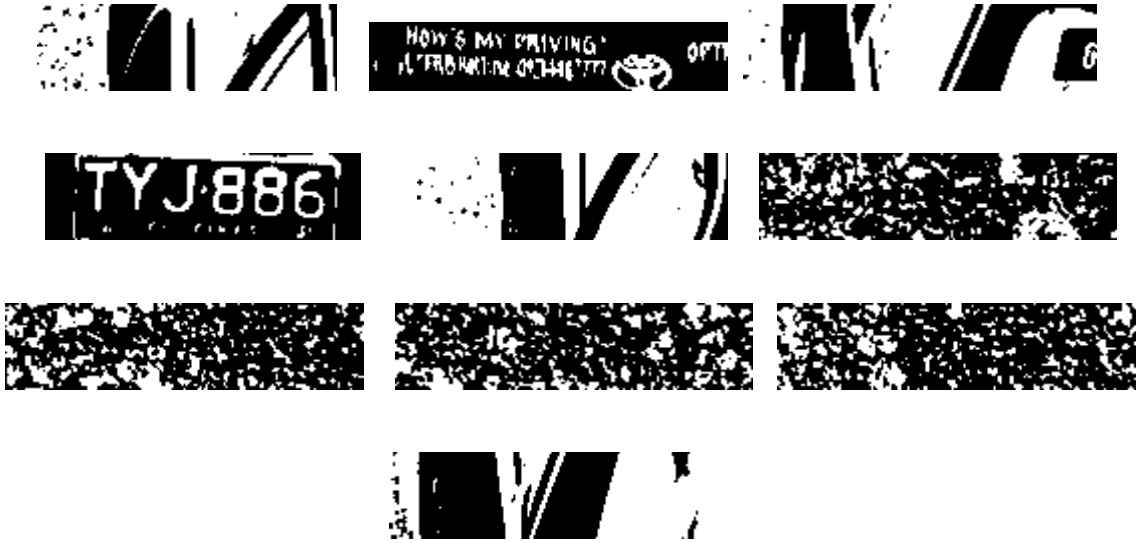

Figure 19. Binarized Plate Candidates

Connected Components:

Figure 20. Connected Components of Plate Candidates

The segmented pieces of each plate candidate are checked if they pass the heuristic factor of having the dimensions of a plate number character. If they possess such characteristic, they are then used in the neural network for recognition.

When the system has recognized a plate number, it displays the characters on the interface of the system.

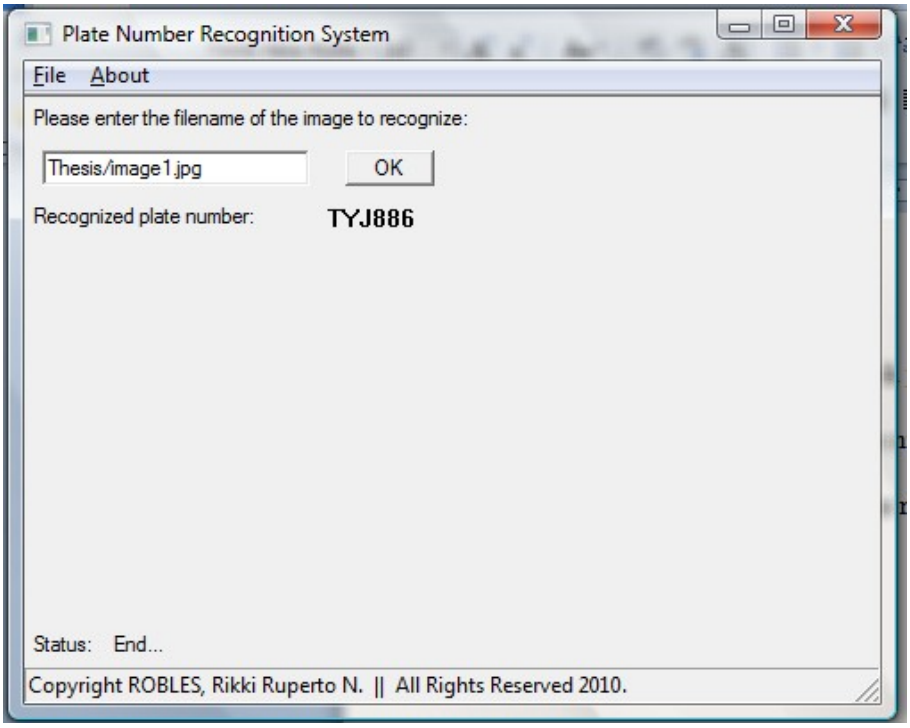

Figure 21. Final Output

VI. Discussion

As a stand-online system, the Plate Number Recognition System theoretically has no particulars users. What it is basically is system that takes in an image as an input and outputs the recognized plate number within that image. The system employs different algorithms associated with computer vision and neural networks.

These algorithms are particularly applied to the input image. First, the image is converted into a grayscale image. This is done by averaging the Red, Green and Blue aspects of each pixel of the image. After undergoing this process, the grayscaled image passes on the second derivative approximation function to detecting edges. Binarizing the edge-detected image will be the next step. With the binarized image, the running-sum algorithm is used ten (10) times to the image to acquire the 10 plate number candidates.

Each plate number candidate's coordinates are used to extract them from the grayscaled image. Having obtained the plate number candidates, their contrast will all be improved so as to help in the segmentation process of the system.

Iterating through each plate number candidate, the characters within them are mined and are compared to some heuristic test in determining if a character is a plate number character or not. If a particular plate candidate has passed this test, its six (6) characters representing the six characters of a normal Philippine plate number will be fully extracted from the image. Once this happens, the characters are used as input for the neural network for pattern recognition.

The system actually utilizes two (2) neural networks – both trained using the backpropagation technique. The first neural network is a Letter – character recognizer. It basically just recognizes the first three (3) characters of the plate number since it's the area wherein all characters are pure letters. The other neural network on the other hand is a Digit-character recognizer. If the first neural network works on the first 3 characters of the plate number, this neural network works on the last three characters. These characters are always digits. The rationale behind having 2 different neural networks is for better accuracy.

Once the characters have passed through the neural networks, the final character array output will be shown.

Given the parameters for performance, which is basically if an input is correct (e.g. an upright segmented character that is recognized by the eye as letter 'T') the output should be correct (letter 'T') and if an input is incorrect (e.g. inverted segmented character that is recognized by the eye as letter 'T') the output should be incorrect (letter closest to an inverted 'T' and not the actual letter 'T'), this is how the system executes.

VII. Conclusion

The Plate Number Recognition System is a stand-alone system that is beneficial in traffic law and policy enforcement. Through this, *'beating the red light'* violators are identified electronically. This ensures minimal, if not zero, contact between enforcers and erring motorists. Its implication is it lessens the chance for corruption to occur.

The use of multiple plate number candidates greatly helped in localizing the correct plate number within the image. Given that the input image is a large-scaled one, it is but inevitable that there will be instances when locating the correct plate number would need more than two, maybe three guesses. And that's where the use of 10 plate number candidates comes in handy.

The wide set of training patterns for both the letters and the digits is also a factor as to how the Plate Number Recognition System fared.

Although the system has proven to recognize plate numbers, it still has its limitations in that the performance of the system is largely dependent on the quality of the image captured by the camera. If the picture has poor lighting or greatly blurred, expect the system to fare low on the accuracy.

VIII. Recommendation

Much of the improvements that can be carried out to the system would be on the implementation of the neural network and the image-processing methods.

A larger set of training patterns for the neural network would surely improve the capability of the system to learn and identify correctly the segmented pixels of the image and thus produce more accurate readings. This would include putting in patterns for less legible characters for instances such as dilapidated plate numbers (not extreme deterioration of quality).

Image enhancers can be added into the mix as well to further improve the likelihood of getting the area of the plate number as a candidate during the processing. This would also help in segmenting the characters within the plate number candidates.

IX. Bibliography

1. "Profile and History of the Metropolitan Government". The Official Website of the Metropolitan Manila Development Authority. http://www.mmda.gov.ph/history.html

2. Velappa Ganapathy and Wen Lik Dennis Lui, "A Malaysian Vehicle License Plate Localization and Recognition System," <http://www.iiisci.org/journal/CV$/sci/pdfs/S985FYB.pdf>.

3. Serkan Ozbay and Ergun Ercelebi, "Automatic Vehicle Identification by Plate Recognition," World Academy of Science, Engineering and Technology. September 2005. <http://www.waset.org/journals/waset/v9/v9-41.pdf>.

4. Cesar Garcia-Osorio, et al. "License Plate Number Recognition, New Heuristics and a Comparative Study of Classifiers," <http://pisuerga.inf.ubu.es/juanjo/bib2html/e-documents/publs/icinco08.pdf>.

5. Osslan Osiris Vergara Villegas, et al. "License Plate Recognition Using a Novel Fuzzy Multilayer Neural Network," International Journal of Computers 3. 1 (2009).

6. A. Broumandnia and M. Fathi, "Application of Pattern Recognition for Farsi License Plate Recognition," ICGST-GVIP Journal 5, 2 (2005).

7. Felix Arya and Iping Supriana Suwardi, "License Plate Recognition System for Indonesian Vehicles," <http://repository.gunadarma.ac.id:8000/B-87_456.pdf>.

8. I-Chen Tsai, et al., “Recognition of Vehicle License Plates from a Video Sequence,” IAENG International Journal of Computer Science 36. 1 (2009).

9. Hsien-Chu Wu, et al., “A License Plate Recognition System in E-Government,” <http://se2.isn.ch/serviceengine/Files/EINIRAS/10523/ichaptersection_singledocument/F372B546-C6FF-4BAA-B151-8AFC6E6C3EE0/en/doc_10553_259_en.pdf>.

10. Ales Janota, et al., “Attributes Selection for License Plate Recognition Based on Decision Trees,” Acta Electrotechnica et Informatica 4, 5 (2005).

11. N. Vasquez, et al., “Automatic System for Localization and Recognition of Vehicle Plate Numbers,” Journal of Applied Research and Technology 1, 1 (Apr. 2003): 63-77

12. Ondrej Martinsky, “Algorithmic and Mathematical Principles of Aumotamitc Number Plate Systems,” http://javaanpr.sourceforge.net/anpr.pdf

13. “Electronic Records Management Guidelines, Glossary,” http://www.mnhs.org/preserve/records/electronicrecords/erglossary.html

14. “Introduction to Neural Networks,” Neural Networks. http://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html#Introduction%20to%20neural%20networks

15. System for Copyright Protection – Glossary. http://www.igd.fraunhofer.de/igd-a8/syscop/glossary.html

16. “The Sobel Operator Used in Image Processing,” Concise Articles on Science, Math and Technology. http://www.starkeffects.com/sobel-operator.htm

17. “Connected Components Labelling,” Image Analysis – Connected Components Labelling. http://homepages.inf.ed.ac.uk/rbf/HIPR2/label.htm

18. Juan Miguel J. Bawagan, et al., “License Plate Localization for Difficult Cases.”

19. Abdou, I. E., et al., “Computer Vision,” http://homepages.inf.ed.ac.uk/rbf/BOOKS/BANDB/LIB/bandb1.pdf

20. “Red Light Camera,” <http://en.wikipedia.org/wiki/Red_light_camera>

21. “How Red-light Cameras Work,” How Stuff Works. <http://auto.howstuffworks.com/car-driving-safety/safety-regulatory-devices/red-light-camera1.htm>

X. Appendix

Source code: pnrsofficial.cpp

```
/* ROBLES, Rikki Ruperto N. 2006-27713
*  -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
*    PLATE NUMBER RECOGNITION SYSTEM
*  -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
*/

#include "image.h"
#include "windows.h"
#include "resource.h"
#include "stdio.h"
#include "stdlib.h"
#include "ctime"
#include "jpegio.h"
#include "filter.h"
#include "math.h"
#include "binary.h"
#include "iostream.h"
#include "fstream.h"
#include <algo.h>

#define ID_FILE_EXIT 9001
#define ID_STUFF_ABOUT 9002
#define IDC_MAIN_LIST    101
#define IDC_MAIN_STATUS          102
#define IDC_MAIN_BUTTON          103
#define IDC_MAIN_TEXT  104
#define IDC_TEXT             105
#define IDC_MAIN_EDIT   106
#define IDC_TEXT2            107
#define IDC_TEXT3            114
#define IDC_TEXT4            115
#define IDC_TEXT_1           108
#define IDC_TEXT_2           109
#define IDC_TEXT_3           110
#define IDC_TEXT_4           111
#define IDC_TEXT_5           112
#define IDC_TEXT_6           113

const char g_szClassName[] = "myWindowClass";
HWND hwnd;

// GLOBAL VARIABLES
RGBImage original, img, edgeVisual, gradientVisual,
plate[10], binarizedPlate, characters[6];
int edgeVisualArray[2500][1662];
int platesCoor[10][2], adjustedPlates[10][4], plate_left,
plate_top, plate_right, plate_bottom;
int width_bound = 180;
int height_bound = 40;
int howManyPlates = 0;
// END GLOBAL VARIABLES
```

```
// FUNCTIONS
void edge_detect(char*);
void improve_contrast(int);
void running_sum(int);
void bound(int, int, int, int, int);
int get_center(int, int, int, int, char);
void find_best_plate();
void adjust_bounds(int);
void draw_bound(int, int, int, int, int);
void choose_plate();
void create_plate(int);


void binarize_plate(int);
void connected_components(Image<unsigned char>,
int);
void character_recognition();
// END FUNCTIONS


// CLASS FOR NEURAL NETWORK
class NN{
        private:
                double *input;
                double *output;
            double *hidden;
            int numberInput;
            int numberHidden;
            int numberOutput;
            double **weightInput2Hidden;
            double **weightHidden2Output;
            double *thresholdHidden;
            double *thresholdOutput;
            double *errorOutput;
            double *errorHidden;
            double *errorGradientOutput;
            double *errorGradientHidden;
            double **weightCorrectionI2H;
            double **weightCorrectionH2O;
            double learningRate;
            double momentum;
        public:
                NN(int numberInput, int numberOutput,
int numberHidden);
                ~NN();
            int calculate(int *input);
            void feedforward(double *input);
            char recognize(double *input, int which);
            void load(int which);
};


NN::NN(int numberInput, int numberOutput, int
numberHidden){
   // allocate memory
   this->numberInput = numberInput;
   this->numberHidden = numberHidden;
   this->numberOutput = numberOutput;
   input = new double[numberInput];
   output = new double[numberOutput];
   hidden = new double[numberHidden];
   weightInput2Hidden = new double*[numberInput];
   weightHidden2Output = new
double*[numberHidden];
   thresholdHidden = new double[numberHidden];
   thresholdOutput = new double[numberOutput];
   errorOutput = new double[numberOutput];
   errorHidden = new double[numberHidden];
   errorGradientOutput = new double[numberOutput];
   errorGradientHidden = new double[numberHidden];
```

```
    weightCorrectionI2H = new double*[numberInput];

    weightCorrectionH2O = new
double*[numberHidden];



    int x,y;

    for(x = 0; x < numberInput; x++){

        weightInput2Hidden[x] = new
double[numberHidden];

        weightCorrectionI2H[x] = new
double[numberHidden];

    }

    for(x = 0; x < numberHidden; x++){

        weightHidden2Output[x] = new
double[numberOutput];

        weightCorrectionH2O[x] = new
double[numberOutput];

    }



    // assign random values to weights and threshold

    srand((unsigned)(time(NULL)));

    for(x = 0; x < numberInput; x++){

        for(y = 0; y < numberHidden; y++){

            weightInput2Hidden[x][y] = (double)(rand())/
(RAND_MAX/2)-1;

            weightCorrectionI2H[x][y] = 0.0;

        }

    }

    for(x = 0; x < numberHidden; x++){

        for(y = 0; y < numberOutput; y++){

            weightHidden2Output[x][y] = (double)(rand())/
(RAND_MAX/2)-1;

            weightCorrectionH2O[x][y] = 0.0;

        }

    }

    for(x = 0; x < numberHidden; x++){

        thresholdHidden[x] = (double)(rand())/
(RAND_MAX/2)-1;

    }

    for(x = 0; x < numberOutput; x++){

        thresholdOutput[x] = (double)(rand())/
(RAND_MAX/2)-1;

    }



    learningRate = 0.1;

    momentum = 0.9;

}



NN::~NN(){

    // allocate memory

    delete input;

    delete output;

    delete hidden;

    delete [] weightInput2Hidden;

    delete [] weightHidden2Output;

    delete thresholdHidden;

    delete thresholdOutput;

    delete errorOutput;

    delete errorHidden;

    delete errorGradientOutput;

    delete errorGradientHidden;

}



void NN::feedforward(double *input){ //feedforward
pass

    int x,y;

    // assign input values into input layer

    for(x = 0; x < numberInput; x++){

        this->input[x] = input[x];

    }

    double temp1, temp2;

    // propagate input pattern to hidden layer

    for(x = 0; x < numberHidden; x++){

        temp1 = 0.0; temp2 = 0.0;
```

```
        for(y = 0; y < numberInput; y++){
            temp1 += this->input[y] *
weightInput2Hidden[y][x];
        }
        temp2 = temp1 - thresholdHidden[x];
        hidden[x] = 1.0/(1.0+exp(-temp2));
    }
    //propagate hidden layer values to output layer
    temp1 = 0; temp2 = 0;
    for(x = 0; x < numberOutput; x++){
        temp1 = 0.0; temp2 = 0.0;
        for(y = 0; y < numberHidden; y++){
            temp1 += hidden[y] * weightHidden2Output[y]
[x];
        }
        temp2 = temp1 - thresholdOutput[x];
        output[x] = 1.0/(1.0+exp(-temp2));
    }
}


char NN::recognize(double *input, int which){ //
recognize the input pattern
    feedforward(input);
    double max = -1000000;
    int out;
    int x;
    for(x = 0; x < numberOutput; x++){
        if(max < output[x]){
            max = output[x];
            out = x;
        }
    }
    if(which == 1){
        if(out == 0){
                                if(max < 0.01) return 'X';
                                return 'A';
                }
        else if(out == 1) return 'B';
        else if(out == 2) return 'C';
        else if(out == 3) return 'D';
        else if(out == 4) return 'E';
        else if(out == 5){
                                if(max < 0.1) return 'E';
                                return 'F';
                }
        else if(out == 6) return 'G';
        else if(out == 7) return 'H';
        else if(out == 8){
                                if(max < 0.1) return 'T';
                                return 'I';
                }
        else if(out == 9) return 'J';
        else if(out == 10){
                                if(max < 0.1) return 'Y';
                                return 'K';
                }
        else if(out == 11) return 'L';
        else if(out == 12) return 'M';
        else if(out == 13) return 'N';
        else if(out == 14) return 'O';
        else if(out == 15) return 'P';
        else if(out == 16) return 'Q';
        else if(out == 17) return 'R';
        else if(out == 18) return 'S';
        else if(out == 19) return 'T';
        else if(out == 20) return 'U';
        else if(out == 21) return 'V';
        else if(out == 22) return 'W';
        else if(out == 23) return 'X';
```

```
        else if(out == 24){
                                if(max < 0.1) return 'X';
                                return 'Y';
                        }
        else if(out == 25) return 'Z';
    }else if(which == 2){
        if(out == 0) return '0';
        else if(out == 1) return '1';
        else if(out == 2) return '2';
        else if(out == 3) return '3';
        else if(out == 4) return '4';
        else if(out == 5) return '5';
        else if(out == 6) return '6';
        else if(out == 7) return '7';
        else if(out == 8) return '8';
        else if(out == 9) return '9';
    }
}


void NN::load(int which){ // load the weights for the
neural network for letters and digits
    if(which == 1){
        fstream file_lo("Letters11.txt",ios::in);
        int x,y;
        for(x = 0; x < numberInput; x++){
            for(y = 0; y < numberHidden; y++){
                file_lo >> weightInput2Hidden[x][y];
            }
        }
        for(x = 0; x < numberHidden; x++){
            for(y = 0; y < numberOutput; y++){
                file_lo >> weightHidden2Output[x][y];
            }
        }
        for(x = 0; x < numberHidden; x++){
            file_lo >> thresholdHidden[x];
        }
        for(x = 0; x < numberOutput; x++){
            file_lo >> thresholdOutput[x];
        }
        file_lo.close();
    }else if(which == 2){
        fstream file_lo("Digits7.txt",ios::in);
        int x,y;
        for(x = 0; x < numberInput; x++){
            for(y = 0; y < numberHidden; y++){
                file_lo >> weightInput2Hidden[x][y];
            }
        }
        for(x = 0; x < numberHidden; x++){
            for(y = 0; y < numberOutput; y++){
                file_lo >> weightHidden2Output[x][y];
            }
        }
        for(x = 0; x < numberHidden; x++){
            file_lo >> thresholdHidden[x];
        }
        for(x = 0; x < numberOutput; x++){
            file_lo >> thresholdOutput[x];
        }
        file_lo.close();
    }
}


// THE WINDOWS 32 API PROCEDURE

LRESULT CALLBACK WndProc(HWND hwnd, UINT msg,
WPARAM wParam, LPARAM lParam)

{
```

```
    HWND hEdit;

  switch(msg)

  {

                case WM_CREATE:{

                        HMENU hMenu, hSubMenu;

          HICON hIcon, hIconSm;



          hMenu = CreateMenu();



          hSubMenu = CreatePopupMenu();

          AppendMenu(hSubMenu, MF_STRING,
ID_FILE_EXIT, "E&xit");

          AppendMenu(hMenu, MF_STRING |
MF_POPUP, (UINT)hSubMenu, "&File");



          hSubMenu = CreatePopupMenu();

          AppendMenu(hSubMenu, MF_STRING,
ID_STUFF_ABOUT, "&Author");

          AppendMenu(hMenu, MF_STRING |
MF_POPUP, (UINT)hSubMenu, "&About");



          SetMenu(hwnd, hMenu);



                        //status bar

                        HWND hStatus =
CreateWindowEx(0, STATUSCLASSNAME, NULL,

                      WS_CHILD | WS_VISIBLE |
SBARS_SIZEGRIP, 0, 0, 0, 0,

                      hwnd, (HMENU)IDC_MAIN_STATUS,
GetModuleHandle(NULL), NULL);

                        SendMessage(hStatus,
SB_SETTEXT, 0, (LPARAM)"Copyright ROBLES, Rikki
Ruperto N.  ||  All Rights Reserved 2010.");



                        //ok button

                        HWND hwndButton =
CreateWindowEx(0, "BUTTON", "OK", WS_TABSTOP |
WS_VISIBLE | WS_CHILD | BS_PUSHBUTTON,
                                                180, 30, 50, 20,
hwnd, (HMENU)IDC_MAIN_BUTTON,
GetModuleHandle(NULL), NULL);



                        //static texts

                        HWND hText =
CreateWindowEx(0, "STATIC", "", WS_CHILD |
WS_VISIBLE | SS_LEFT, 5,5,500,20, hwnd,
(HMENU)IDC_MAIN_TEXT,

      GetModuleHandle(NULL), NULL);

                        SendMessage(hText,
WM_SETTEXT, 0, (LPARAM)"Please enter the filename of
the image to recognize:");



                        HWND hText0 =
CreateWindowEx(0, "STATIC", "", WS_CHILD |
WS_VISIBLE | SS_LEFT, 5,60,300,20, hwnd,
(HMENU)IDC_TEXT2,

      GetModuleHandle(NULL), NULL);

                        SendMessage(hText0,
WM_SETTEXT, 0, (LPARAM)"Recognized plate
number:");



                        HWND hTextstat =
CreateWindowEx(0, "STATIC", "", WS_CHILD |
WS_VISIBLE | SS_LEFT, 5,300,50,20, hwnd,
(HMENU)IDC_TEXT3,

      GetModuleHandle(NULL), NULL);

                        SendMessage(hTextstat,
WM_SETTEXT, 0, (LPARAM)"Status:");



                        HWND hTextstat1 =
CreateWindowEx(0, "STATIC", "", WS_CHILD |
WS_VISIBLE | SS_LEFT, 50,300,300,20, hwnd,
(HMENU)IDC_TEXT4,

      GetModuleHandle(NULL), NULL);



                        // where plate numbers will be
contained

                        HWND hText2 =
CreateWindowEx(0, "STATIC", "", WS_CHILD |
WS_VISIBLE | SS_LEFT, 170,60,55,18, hwnd,
(HMENU)IDC_TEXT_1,
```

```
        GetModuleHandle(NULL), NULL);

                                HWND hText3 =
CreateWindowEx(0, "STATIC", "", WS_CHILD |
WS_VISIBLE | SS_LEFT, 170,80,55,18, hwnd,
(HMENU)IDC_TEXT_2,


        GetModuleHandle(NULL), NULL);

                                HWND hText4 =
CreateWindowEx(0, "STATIC", "", WS_CHILD |
WS_VISIBLE | SS_LEFT, 170,100,55,18, hwnd,
(HMENU)IDC_TEXT_3,


        GetModuleHandle(NULL), NULL);

                                HWND hText5 =
CreateWindowEx(0, "STATIC", "", WS_CHILD |
WS_VISIBLE | SS_LEFT, 170,120,55,18, hwnd,
(HMENU)IDC_TEXT_4,


        GetModuleHandle(NULL), NULL);

                                HWND hText6 =
CreateWindowEx(0, "STATIC", "", WS_CHILD |
WS_VISIBLE | SS_LEFT, 170,140,55,18, hwnd,
(HMENU)IDC_TEXT_5,


        GetModuleHandle(NULL), NULL);

                                HWND hText7 =
CreateWindowEx(0, "STATIC", "", WS_CHILD |
WS_VISIBLE | SS_LEFT, 170,160,55,18, hwnd,
(HMENU)IDC_TEXT_6,


        GetModuleHandle(NULL), NULL);



                                //edit

                                HWND hEdit2 =
CreateWindowEx(WS_EX_CLIENTEDGE, "EDIT", "",
WS_CHILD | WS_VISIBLE ,

                                        10, 30, 150, 20,
hwnd, (HMENU)IDC_MAIN_EDIT,
GetModuleHandle(NULL), NULL);



                                //font

                                HFONT hfDefault;

                                hfDefault =
GetStockObject(DEFAULT_GUI_FONT);

                                SendMessage(hEdit,
WM_SETFONT, (WPARAM)hfDefault,
MAKELPARAM(FALSE, 0));

                                SendMessage(hEdit2,
WM_SETFONT, (WPARAM)hfDefault,
MAKELPARAM(FALSE, 0));

                                SendMessage(hwndButton,
WM_SETFONT, (WPARAM)hfDefault,
MAKELPARAM(FALSE, 0));

                                SendMessage(hText,
WM_SETFONT, (WPARAM)hfDefault,
MAKELPARAM(FALSE, 0));

                                SendMessage(hText0,
WM_SETFONT, (WPARAM)hfDefault,
MAKELPARAM(FALSE, 0));

                                SendMessage(hTextstat,
WM_SETFONT, (WPARAM)hfDefault,
MAKELPARAM(FALSE, 0));

                                SendMessage(hTextstat1,
WM_SETFONT, (WPARAM)hfDefault,
MAKELPARAM(FALSE, 0));

                                }

                                break;

                        case WM_SIZE:{



                                HWND hStatus;

                                RECT rcStatus;

                                int statusHeight;



                                hStatus = GetDlgItem(hwnd,
IDC_MAIN_STATUS);

                           SendMessage(hStatus, WM_SIZE, 0,
0);



                           GetWindowRect(hStatus, &rcStatus);

                           statusHeight = rcStatus.bottom -
rcStatus.top;

                                }

                                break;

                        case WM_COMMAND:{

                                switch(LOWORD(wParam)){

                                        case
ID_FILE_EXIT: // exit button


      PostMessage(hwnd, WM_CLOSE, 0, 0);
```

```
                                        break;
                                        case
ID_STUFF_ABOUT:{ // about button

      MessageBox(hwnd,"The Author of this program
is\r\nROBLES,Rikki","Author",MB_OK |
MB_ICONINFORMATION);
                                        }
                                        break;
                                        case
IDC_MAIN_BUTTON:{ // ok button which would start the
recognition process

                                                int len =
GetWindowTextLength(GetDlgItem(hwnd,
IDC_MAIN_EDIT));
                                          if(len > 0)
                                          {
                                             int i;
                                             char* buf;


      // get file path from the input box
                                             buf =
(char*)GlobalAlloc(GPTR, len + 1);

GetDlgItemText(hwnd, IDC_MAIN_EDIT, buf, len + 1);



      int x;
      //plate localization
      howManyPlates = 0;
      SetDlgItemText(hwnd, IDC_TEXT_1, "");
      SetDlgItemText(hwnd, IDC_TEXT_2, "");
      SetDlgItemText(hwnd, IDC_TEXT_3, "");
      SetDlgItemText(hwnd, IDC_TEXT_4, "");
      SetDlgItemText(hwnd, IDC_TEXT_5, "");
      SetDlgItemText(hwnd, IDC_TEXT_6, "");

edge_detect(buf);
      GlobalFree((HANDLE)buf);
                                          for(x = 1; x <
11; x++)
            running_sum(x);
                                          for(x = 1; x <
11; x++)
            adjust_bounds(x);
                                          choose_plate();


      //character segmentation
      for(x = 0; x < 10; x++)
            binarize_plate(x);


      SetDlgItemText(hwnd, IDC_TEXT4, "End...");


                                                return 0;
                                       }
                                    }
                                    break;
                              }
                              }
                              break;
      case WM_CLOSE:
         DestroyWindow(hwnd);
         break;
      case WM_DESTROY:
```

```
        PostQuitMessage(0);

        break;

      default:

        return DefWindowProc(hwnd, msg, wParam,
lParam);

   }

   return 0;

}



// WINDOWD 32 API MAIN

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE
hPrevInstance,

   LPSTR lpCmdLine, int nCmdShow)

{

   WNDCLASSEX wc;

   MSG Msg;



   // Registering the Window Class

   wc.cbSize        = sizeof(WNDCLASSEX);

   wc.style         = 0;

   wc.lpfnWndProc   = WndProc;

   wc.cbClsExtra    = 0;

   wc.cbWndExtra    = 0;

   wc.hInstance     = hInstance;

   wc.hIcon         = LoadIcon(NULL, IDI_APPLICATION);

   wc.hCursor       = LoadCursor(NULL, IDC_ARROW);

   wc.hbrBackground = (HBRUSH)(COLOR_WINDOW);

   wc.lpszMenuName  = NULL;

   wc.lpszClassName = g_szClassName;

   wc.hIconSm       = LoadIcon(NULL, IDI_APPLICATION);

      wc.lpszMenuName  =
MAKEINTRESOURCE(IDR_MYMENU);



   if(!RegisterClassEx(&wc))

   {

      MessageBox(NULL, "Window Registration Failed!",
"Error!",

         MB_ICONEXCLAMATION | MB_OK);

      return 0;

   }



   // Creating the Window

   hwnd = CreateWindowEx(

      WS_EX_CLIENTEDGE,                    //extended
windows style

      g_szClassName,                       //what kind of
window to create

      "Plate Number Recognition System",       //title

      WS_OVERLAPPEDWINDOW,                 //window
style parameter

      CW_USEDEFAULT, CW_USEDEFAULT, 500, 400,
//window dimension

      NULL, NULL, hInstance, NULL);          //parent
window handle, menu, app instance, pointer



   // checker

   if(hwnd == NULL)

   {

      MessageBox(NULL, "Window Creation Failed!",
"Error!",

         MB_ICONEXCLAMATION | MB_OK);

      return 0;

   }



   // show window and make sure it is correctly drawn

   ShowWindow(hwnd, nCmdShow);

   UpdateWindow(hwnd);



   // The Message Loop

   while(GetMessage(&Msg, NULL, 0, 0) > 0)

   {

      TranslateMessage(&Msg);
```

```
      DispatchMessage(&Msg);
    }
    return Msg.wParam;
}


// FIND THE EDGES OF THE IMAGE
void edge_detect(char s[100]){
  Image<double> meanMask;
  Image<double> grayScale;
  Image<double> filteredIntensity;
  int x,y,height,width;

  // the gradient magnitude threshold separates
  // the significant edges from the non-sgnificant edges in the image
  double gradientMagnitudeThreshold = 30;

  // read the JPEG image
  readJpeg( original, s );
  img = original;
  width = img.width();
  height = img.height();
  edgeVisual.resize(width,height);  // RGB visualization of the significant edges
  gradientVisual.resize(width,height);  // RGB visualization of the gradient
  SetDlgItemText(hwnd, IDC_TEXT4, "JPEG File read...");


  // convert the input image to gray scale
  grayScale.resize( width,height );
  for (x = 0; x < width; x++ )
    for (y = 0; y < height; y++ ){
      grayScale(x,y) = (RED(img(x,y)) + GREEN(img(x,y)) + BLUE(img(x,y))) / 3;
    }

  // smooth the image first to reduce noise
  meanMask.resize(5,5);
  meanMask.setAll( 1.0/25.0 );
  convolveDouble( filteredIntensity, grayScale, meanMask );


  // compute the edge values and convert to binary image
  for (x = 0; x < width; x++) {
    for (y = 0; y < height; y++) {
      int grayValue = grayScale(x-2,y) - (2*grayScale(x,y)) + grayScale(x+2,y);
      gradientVisual(x,y) = COLOR_RGB(grayValue,grayValue,grayValue);
      if(gradientVisual(x,y) < 0) gradientVisual(x,y) *= -1;
      if (RED(gradientVisual(x,y)) < gradientMagnitudeThreshold)
        edgeVisual(x,y) = COLOR_RGB(0,0,0);   // NOT an edge
      else
        edgeVisual(x,y) = COLOR_RGB(255,255,255);   // a significant edge pixel
    }
  }


  // write the visualization of the edges
  writeJpeg( edgeVisual, "Thesis/Output/edges.jpg", 100 );
  SetDlgItemText(hwnd, IDC_TEXT4, "Edge detected...");
  writeJpeg( gradientVisual, "Thesis/Output/gradient.jpg", 100 );
  SetDlgItemText(hwnd, IDC_TEXT4, "Gradient Visual created...");


  for(y = 0; y < edgeVisual.height(); y++){
    for(x = 0; x < edgeVisual.width(); x++){
      edgeVisualArray[x][y] = RED(edgeVisual(x,y));
```

```
    }

  }

}


// FIND THE AREA WITH HIGHEST EDGE
CONCENTRATION

void running_sum(int round){

   int x,y,i;

      SetDlgItemText(hwnd, IDC_TEXT4, "Locating Plate
Candidates.");

   int first_run[edgeVisual.width()][edgeVisual.height() -
height_bound + 1];

   //first run through the picture

   for(y = 0; y < edgeVisual.height() - height_bound +
1; y++){

      int first_run_through = 0;

      for(x = 0; x < edgeVisual.width(); x++){

         first_run_through = edgeVisualArray[x][y];

         for(i = 1; i < height_bound; i++){

            first_run_through += edgeVisualArray[x]
[y+i];

         }

         first_run[x][y] = first_run_through;

      }

   }



   int second_run[edgeVisual.width() - width_bound +
1][edgeVisual.height() - height_bound + 1];

   //second run through the picture

   for(y = 0; y < edgeVisual.height() - height_bound +
1; y++){

      int second_run_through = 0;

      for(x = 0; x < edgeVisual.width() - width_bound +
1; x++){

         second_run_through = first_run[x][y];

         for(i = 1; i < width_bound; i++){

            second_run_through += first_run[x+i][y];

         }

         second_run[x][y] = second_run_through;

      }

   }



   int first_x,first_y;

   int max_value=0;

   //get the x,y of the upper left most part of the area
with largest concentration of edges

   for(y = 0; y < edgeVisual.height() - height_bound; y+
+){

      for(x = 0; x < edgeVisual.width() - width_bound;
x++){

         if(max_value < second_run[x][y]){

            max_value = second_run[x][y];

            first_x = x;

            first_y = y;

         }

      }

   }

   bound(first_x,first_y,width_bound,height_bound,roun
d);



   for(y = first_y; y <= first_y+height_bound; y++){

      for(x = first_x; x <= first_x+width_bound; x++){

         edgeVisualArray[x][y] = 0;

      }

   }

   writeJpeg( img, "Thesis/Output/bounded.jpg", 100 );

}



// GET THE BOUNDS

void bound(int first_x, int first_y, int width_bound, int
height_bound, int round){

      SetDlgItemText(hwnd, IDC_TEXT4, "Locating Plate
Candidates..");

   int center_x = get_center(first_x, first_y,
width_bound, height_bound, 'x');
```

```
    int center_y = get_center(first_x, first_y,
width_bound, height_bound, 'y');

    int newX = center_x - (width_bound/2);

    int newY = center_y - (height_bound/2);



    platesCoor[round - 1][0] = newX;

    platesCoor[round - 1][1] = newY;



    draw_bound(first_x,first_y,width_bound,height_bound
,1);

    draw_bound(newX,newY,width_bound,height_bound,
2);

}



// GET CENTER OF HIGHEST EDGE CONCENTRATION

int get_center(int x_coor, int y_coor, int width, int
height, char which){

       SetDlgItemText(hwnd, IDC_TEXT4, "Locating Plate
Candidates...");

    int y,x;

    if(which == 'x'){

        int numerator = 0;

        for(x = x_coor; x < x_coor + width; x++){

            int summation = 0;

            for(y = y_coor; y < y_coor + height; y++){

                summation += x*(RED(edgeVisual(x,y)));

            }

            numerator += summation;

        }

        int denominator = 0;

        for(x = x_coor; x < x_coor + width; x++){

            int summation = 0;

            for(y = y_coor; y < y_coor + height; y++){

                summation += (RED(edgeVisual(x,y)));

            }

            denominator += summation;

        }

        return numerator/denominator;

    }else if(which == 'y'){

        int numerator = 0;

        for(x = x_coor; x < x_coor + width; x++){

            int summation = 0;

            for(y = y_coor; y < y_coor + height; y++){

                summation += y*(RED(edgeVisual(x,y)));

            }

            numerator += summation;

        }

        int denominator = 0;

        for(x = x_coor; x < x_coor + width; x++){

            int summation = 0;

            for(y = y_coor; y < y_coor + height; y++){

                summation += (RED(edgeVisual(x,y)));

            }

            denominator += summation;

        }

        return numerator/denominator;

    }

}



// ADJUST THE DIMENSION OF AREA BOUND

void adjust_bounds(int round){

    int y,x,top,bottom,left,right;

    // get top and bottom

    bool isTop = false;

    int bottomcounter = 0;

    for(y = platesCoor[round-1][1]; y <
platesCoor[round-1][1]+height_bound; y++){

        int rowedge = 0;

        if(bottomcounter > 5){

            bottom = y;
```

```
            break;

        }

        for(x = platesCoor[round-1][0]; x <
platesCoor[round-1][0]+width_bound; x++){

            if(RED(edgeVisual(x,y)) != 0){

                rowedge++;

            }

            if(rowedge > 10 && isTop != true) {

                top = y;

                isTop = true;

                break;

            }

            if(rowedge > 10 && isTop == true){

                bottom = y;

            }

        }

        if(rowedge < 10 && isTop == true)
bottomcounter++;

    }

    // get left and right

    bool isLeft = false;

    for(x = platesCoor[round-1][0]; x <
platesCoor[round-1][0]+width_bound; x++){

        int columnedge = 0;

        for(y = platesCoor[round-1][1]; y <
platesCoor[round-1][1]+height_bound; y++){

            if(RED(edgeVisual(x,y)) != 0){

                columnedge++;

            }

            if(columnedge > 4 && isLeft != true){

                left = x;

                isLeft = true;

                break;

            }

            if(columnedge > 4 && isLeft == true){

                right = x;

            }

        }

    }



    for(y = top; y <= bottom; y++){

            img(left,y) = COLOR_RGB(0,0,255);

            img(left+1,y+1) = COLOR_RGB(0,0,255);   //
make the line 2-pixel thick

            img(right,y) = COLOR_RGB(0,0,255);

            img(right+1,y+1) = COLOR_RGB(0,0,255);   //
make the line 2-pixel thick

    }

    for(x = left; x <= right;  x++) {

            img(x,top) = COLOR_RGB(0,0,255);

            img(x+1,top+1) = COLOR_RGB(0,0,255);   //
make the line 2-pixel thick

            img(x,bottom) = COLOR_RGB(0,0,255);

            img(x+1,bottom+1) = COLOR_RGB(0,0,255);  //
make the line 2-pixel thick

    }



    adjustedPlates[round - 1][0] = left;

    adjustedPlates[round - 1][1] = top;

    adjustedPlates[round - 1][2] = right;

    adjustedPlates[round - 1][3] = bottom;



    writeJpeg( img, "Thesis/Output/bounded.jpg", 100 );

}



// DRAWS THE BOUND

void draw_bound(int xcoor, int ycoor, int width, int
height, int color){

    int y,x;

    //draw left and right boundaries

    for(y = ycoor; y <= ycoor+height; y++){

        if (color == 1){
```

```
            img(xcoor,y) = COLOR_RGB(255,128,0);

            img(xcoor+1,y+1) =
COLOR_RGB(255,128,0);   // make the line 2-pixel thick

            img(xcoor+width,y) = COLOR_RGB(255,128,0);

            img(xcoor+width+1,y+1) =
COLOR_RGB(255,128,0);   // make the line 2-pixel thick

        }else if (color == 2){

            img(xcoor,y) = COLOR_RGB(255,0,0);

            img(xcoor+1,y+1) = COLOR_RGB(255,0,0);   //
make the line 2-pixel thick

            img(xcoor+width,y) = COLOR_RGB(255,0,0);

            img(xcoor+width+1,y+1) =
COLOR_RGB(255,0,0);   // make the line 2-pixel thick

        }

    }

    //draw top and bottom boundaries

    for(x = xcoor; x <= xcoor+width;  x++) {

        if (color == 1){

            img(x,ycoor) = COLOR_RGB(255,128,0);

            img(x+1,ycoor+1) =
COLOR_RGB(255,128,0);   // make the line 2-pixel thick

            img(x,ycoor+height) = COLOR_RGB(255,128,0);

            img(x+1,ycoor+height+1) =
COLOR_RGB(255,128,0);  // make the line 2-pixel thick

        }else if(color == 2){

            img(x,ycoor) = COLOR_RGB(255,0,0);

            img(x+1,ycoor+1) = COLOR_RGB(255,0,0);   //
make the line 2-pixel thick

            img(x,ycoor+height) = COLOR_RGB(255,0,0);

            img(x+1,ycoor+height+1) =
COLOR_RGB(255,0,0);  // make the line 2-pixel thick

        }

    }

}


// PICKS THE PLATE CANDIDATE

void choose_plate(){

    int x;

    for(x = 0; x < 10; x++){

            plate_left = adjustedPlates[x][0];

            plate_top = adjustedPlates[x][1]-2;

            plate_right = adjustedPlates[x][2];

            plate_bottom = adjustedPlates[x][3]+2;

                                        create_plate(x);

    }

}


// EXTRACT THE LOCALIZED PLATE CANDIDATE

void create_plate(int counter){

        SetDlgItemText(hwnd, IDC_TEXT4, "Creating Plate
Candidate JPEG File...");

    int y,x;

    plate[counter].resize(plate_right-
plate_left,plate_bottom-plate_top);

    for(y = 0; y < plate_bottom-plate_top; y++){

        for(x = 0; x < plate_right-plate_left; x++){

            plate[counter](x,y) =
original(plate_left+x,plate_top+y);

        }

    }

    char charFileName[100];

        sprintf( charFileName, "Thesis/Output/plate
%d.jpg", counter );

    writeJpeg( plate[counter], charFileName, 100 );

}


// BINARIZE THE EXTRACTED PLATE CANDIDATE

void binarize_plate(int counter){

        SetDlgItemText(hwnd, IDC_TEXT4, "Segmenting
Plate Candidate...");

    Image<unsigned char> binary;


    int intensityThreshold = 100;
```

```
   int height = plate[counter].height();

   int width = plate[counter].width();

   int x,y, gray;


   improve_contrast(counter);


   binary.resize(width, height);

   binary.setAll(0);


   binarizedPlate.resize(width, height);

   binarizedPlate.setAll(COLOR_RGB(0,0,0));


   for (x = 0; x < width;  x++) {

    for (y = 0; y < height; y++) {

      gray = (RED(plate[counter](x,y)) +
GREEN(plate[counter](x,y)) + BLUE(plate[counter]
(x,y))) / 3;

      if (gray < intensityThreshold) {   // if the pixel is
dark

        binary(x,y) = 1;   // this pixel is part of the text

        binarizedPlate(x,y) =
COLOR_RGB(255,255,255);  // in the visualization, it will
be colored black

      }

    }

  }


  connected_components(binary, counter);

}


// FIND THE CONNECTED COMPONENTS IN THE PLATE
CANDIDATE

void connected_components(Image<unsigned char>
binary, int counter){

  ConnectedComponents cc;

  RGBImage outputImage, character;

  Image<unsigned char> component, strucElem;

  int counter1, coor[6];

  char charFileName[30];


  cc.analyzeBinary( binary, EIGHT_CONNECTED );


  outputImage = cc.randomColors();    // show each
component with a different color

  sprintf( charFileName, "Thesis/Output/pieces%d.jpg",
counter );

  writeJpeg( outputImage, charFileName, 100 );

  sprintf( charFileName, "Thesis/Output/binary%d.jpg",
counter );

  writeJpeg( binarizedPlate, charFileName, 100 );


  int charCount = 0;

  for(counter1 = 0; counter1 <
cc.getNumComponents(); counter1++){

     if(charCount < 6){

          int x,y,w,h,j,k;

                strucElem.resize(3,3);   // A square
structuring element

                strucElem.setAll(1);

          component =
cc.getComponentBinary(counter1);

                // dilation fills in gaps

                component =
binaryDilation( component, strucElem, 1,1 );

                strucElem.resize(2,2);   // A square
structuring element

                strucElem.setAll(1);

                // erosion is effective in reducing noise

                component =
binaryErosion( component, strucElem, 1,1 );


          cc.getBoundary(counter1,x,y,w,h);

          character.resize(w,h);


          if((character.width() >= 3 &&
character.width() < 20) &&
```

```
                (character.height() < 28 &&
character.height() >= 17)){

                                SetDlgItemText(hwnd,
IDC_TEXT4, "Segmenting Characters...");

            for(j = 0; j < w; j++){

               for(k = 0; k < h; k++){

                  character(j,k) = (component(j,k) == 0)
? COLOR_RGB(255,255,255) : COLOR_RGB(0,0,0);

               }

            }

            sprintf( charFileName,
"Thesis/Output/Plates/%04d.jpg", counter );

            writeJpeg( character, charFileName, 100 );



            coor[charCount] = x;

            characters[charCount++] = character;

         }

      }

   }

   if(charCount == 6){

      int x, n = 5;

      bool unsorted;

      do{

         unsorted = false;

         for(x = 0; x < n; x++){

            int temp1;

            RGBImage temp2;

            if(coor[x] > coor[x+1]){

               temp1 = coor[x];

               coor[x] = coor[x+1];

               coor[x+1] = temp1;

               temp2 = characters[x];

               characters[x] = characters[x+1];

               characters[x+1] = temp2;

               unsorted = true;

            }

         }

         n--;

      }while(unsorted);

      character_recognition();

   }

}



// IMPROVE THE CONTRAST OF THE GRAYSCALE IMAGE

void improve_contrast(int counter){

   SetDlgItemText(hwnd, IDC_TEXT4, "Improving
Contrast of Plate Candidate...");

   int x,y, height,width;

   int pix;

   int numSamples, histSum;

   Image<unsigned char> gray;

   int h;



   unsigned char Ymap[256];  // contains the new values
of each gray value in 0..255

   unsigned char newGray;

   char outputFileName[100];




   height = plate[counter].height();

   width  = plate[counter].width();



   // (startX,startY) and (endX,endY) define the corners
of the region that will be enhanced

   int startX = 0;

   int startY = 0;

   int endX = width;

   int endY = height;



   int percent = 100;  // amount of contrast
enhancement  (maximum is 100)
```

```
  gray.resize( width,height );


  // compute the gray scale of the sub-image
  for (y = startY; y < endY; y++) {
    for (x = startX; x < endX;  x++) {
      pix = plate[counter](x,y);
      gray(x,y) = (RED(pix) + GREEN(pix) + BLUE(pix)) /
3;
    }
  }


  Image<int> hist;
  double jth_perc, pth_perc;
  double j, p, b, m;
  double x_intercept, y_intercept;
  int ctr = 0;


  hist.resize(256,1);   // hist(v,0) is the frequency of
intensity v
  hist.setAll(0);
  for (x = 0; x < width; x++)   // for each pixel
    for (y = 0; y < height; y++)
      hist(gray(x,y),0)++;


  j = (width*height)*0.05;
  p = (width*height)*0.90;


  for(x=0; x<256; x++){
     ctr = ctr + hist(x,0);
                    if(ctr>=j){
                      jth_perc = (double)x;
                      break;
     }
  }
  ctr=0;
  for(x=0; x<256; x++){
     ctr = ctr + hist(x,0);
                    if(ctr>=p){
                      pth_perc = (double)x;
                      break;
     }
  }
  m = (255-0)/(pth_perc-jth_perc);
  b = 255-(m*pth_perc);


   plate[counter].resize(width, height);
   for (x = 0; x < width; x++) {  // for each pixel
   for (y = 0; y < height; y++){
                    if(gray(x,y)<=jth_perc)
                                plate[counter]
(x,y)=COLOR_RGB(0,0,0);


                    else if(gray(x,y)>=pth_perc)
                                        plate[counter]
(x,y)=COLOR_RGB(255,255,255);


                    else {
            x_intercept = gray(x,y);
                    y_intercept = m*x_intercept + b;


                    plate[counter]
(x,y)=COLOR_RGB((int)y_intercept,(int)y_intercept,
(int)y_intercept);
                     }
                    }
        }


  // write the output image to a JPEG file
  char charFileName[30];
  sprintf( charFileName, "Thesis/Output/histeq%d.jpg",
counter );
```

```
  writeJpeg( plate[counter], charFileName, 100 );

}



// RECOGNIZE THE CHARACTERS

void character_recognition(){

   SetDlgItemText(hwnd, IDC_TEXT4, "Recognizing
Characters...");

       //character recognition

       RGBImage recog;

       char toOutput[7];

       int x, y, z;

       double input[6][25*15];

       for(x = 0; x < 6; x++){

                   recog = characters[x];

                   int counter = 0;

                   for(y = 0; y < 25; y++){

                               for(z = 0; z < 15; z++){

                                           if(z > recog.width()
|| y > recog.height()){

                                                       input[x]
[counter++] = 0;

                                           }

                                           else
if(RED(recog(z,y)) > 80){

                                                       input[x]
[counter++] = 0;

                                           }else{

                                                       input[x]
[counter++] = 1;

                                           }

                               }

                   }

       }



       NN Letters(25*15,26,53);

       NN Digits(25*15,10,53);

       Letters.load(1);

       Digits.load(2);

       for(x = 0; x < 3; x++){

                   toOutput[x] =
Letters.recognize(input[x],1);

       }

       for(x = 3; x < 6; x++){

                   toOutput[x] =
Digits.recognize(input[x],2);

       }

       toOutput[6] = '\0';

       SetDlgItemText(hwnd, IDC_TEXT4, "Characters
Recognized!");

       if(howManyPlates == 0){

                   SetDlgItemText(hwnd, IDC_TEXT_1,
toOutput);

                   howManyPlates++;

       }else if(howManyPlates == 1){

                   SetDlgItemText(hwnd, IDC_TEXT_2,
toOutput);

                   howManyPlates++;

       }else if(howManyPlates == 2){

                   SetDlgItemText(hwnd, IDC_TEXT_3,
toOutput);

                   howManyPlates++;

       }else if(howManyPlates == 3){

                   SetDlgItemText(hwnd, IDC_TEXT_4,
toOutput);

                   howManyPlates++;

       }else if(howManyPlates == 4){

                   SetDlgItemText(hwnd, IDC_TEXT_5,
toOutput);

                   howManyPlates++;

       }else if(howManyPlates == 5){

                   SetDlgItemText(hwnd, IDC_TEXT_6,
toOutput);

       }

}
```

XI. Acknowledgement

It took one long semester for me to finish my thesis and defend it in front of a panel and another 2 more years to finally complete my paper. Much of my inspiration and drive in accomplishing these feats can be thanked to a lot of people.

I give my deepest gratitude to my family. I never really was open with them with matters of school, but came thesis time and I was all about how hard it was for me to undertake. Fortunately, my parents supported me through the most generous ways: bringing me late at night to UPM to have my consultation with my adviser, having our driver stay with me the whole day

for yet another consultation at the most unfamiliar and farthest point of Metro Manila I can imagine, and most especially and abundant of all, their unconditional care. Seeing me beaten and depressed after a lengthy discussion with my adviser might have broken their hearts, but their words of encouragement and assurance never ceased. They gave me motivation to go through with all the challenges stacked against me.

Being left behind was a very dark and ill-fated idea for me, and it still is. The thought that all my hardships throughout the year would not bear fruit in the guise of my graduation would have been the death of my self-confidence and the usher of self-doubt. But my friends were there thankfully. They were like *pressure-to-graduate* personified, for two good reasons. One – because my best friend was the first one ever from our batch to defend his thesis successfully. Second – everyone started following suit. And what was I to do but join their ranks. So I relentlessly coded my way through every obstacle to enjoy the same freedom they were relishing. I could not thank them enough for providing me the much needed push to brave this ordeal and finish this surprisingly as one of the best.

And to the two professors who made sure that I present nothing but excellence on my special problem – Sir Baes and Sir Solano – I bid them my sincere appreciation. It was not an easy ride for me and my adviser before we reached the eve of my defence, but he was there to guide and teach me the level of work that was presentable and credible enough for my thesis to be accepted by the panel. The defence itself was not an easy one as well. It was, at the very least, nerve-wracking and challenging. For which I thank Sir Solano. Not only did he adopt me during those few hours of gruelling defence as his advisee, but he extended this up to 2 years until I finished my revisions and was ready to submit all my requirements.

Once again, thanks to these people for making my college life memorable and fulfilling.