HANDWRITING RECOGNITION ON LIBRARY BOOK LABEL

By

Leow Ee Wen

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF COMPUTER SCIENCE (HONS)

Faculty of Information and Communication Technology

(Perak Campus)

MAY 2013

Table of Content

DECLARATION OF ORIGINALITY

I declare that this report entitled “Handwriting Recognition on Library Book Label” is my own work except as cited in the references. The report has not been accepted for any degree and is not being submitted concurrently in candidature for any degree or other award.

Signature : _____

Name : _____

Date : _____

Acknowledgement

First of all, I would like to express my special thanks to my supervisor, Dr. Ng Yen Kaow for giving me the opportunity to complete this project. Besides, I would like to give a grateful thank to Dr. Cheng Wai Khuen for being my project moderator. This was the first time I have myself involved in pattern recognition field and it is able to helps me in expanding my knowledge and experience.

Secondly, I would also like to thanks my friends who helping me so that I can finish the project within time. I would like to thank to Dr. Ng once again for providing me a lot of advices and guidelines. Lastly I would thank to my family which gives me the support during the development process.

Thanks again to all who helped me.

Abstracts

This project aims to design a character recognizer for use in a smart phone application to identify misplaced books on a library shelf. The application works as follows: The user first takes a photo of books on a library shelf. The application then extracts the library labels and performs character recognition on the label texts to obtain the call numbers. If a book is misplaced, then its call number will be out of sequence with the others, and the application can alert the user to such a situation. Such an application will be useful for librarians to perform inventory checks. There are two important technical difficulties to overcome, namely, image segmentation and handwriting recognition. This project studies the latter. Two important aspects are required of the handwriting recognizer for the mobile application to be usable: (1) The recognizer should be able to analyze each photo in the fraction of a second; (2) The recognizer should achieve near-perfect accuracy. In this study, a variety of handwriting recognition techniques are surveyed. Then, a few suitable techniques are chosen, modified, and combined to result in a highly accurate and efficient system. The resultant system, when tested using our database of handwritten characters from library labels, showed near-perfect accuracy.

List of Figures

List of Abbreviations

ANN	Artificial Neural Network
KNN	K-Nearest Neighbours
Matlab	MATrix LABoratory
OCR	Optical Character Recognition
OpenCV	Open Source Computer Vision Library
SDLC	System Development Life Cycle
UTAR	University Tunku Abdul Rahman

CHAPTER 1: INTRODUCTION

1.1 Project Overview

Optical character, or handwritten text recognition has been a popular field of research for many years. One reason for this popularity is in its usefulness in many tasks such as converting scanned documents into digital form, or in interpreting written input from an electronic device (e.g. electronic dictionary, smart phones).

It is often the case for a new task to either require significant modification of available optical character recognition techniques, or the development of entirely new methods, to accomplish. For example, in identifying handwritten text from a touch pad, information such as the order of pen strokes is available to the recognizer. On the other hand, in the task of determining words from papers or other source material, no such information is available. The former type of recognition is known as online recognition, while the latter is known as offline recognition in the literature.

In this project II consider an offline recognition problem which arises out of the task of identifying misplaced books on library shelves. In this task, one is given a picture of a library shelf with books. Each book is tagged with a label printed with a code called call number. Books on the library shelf are to be ordered by these call numbers.

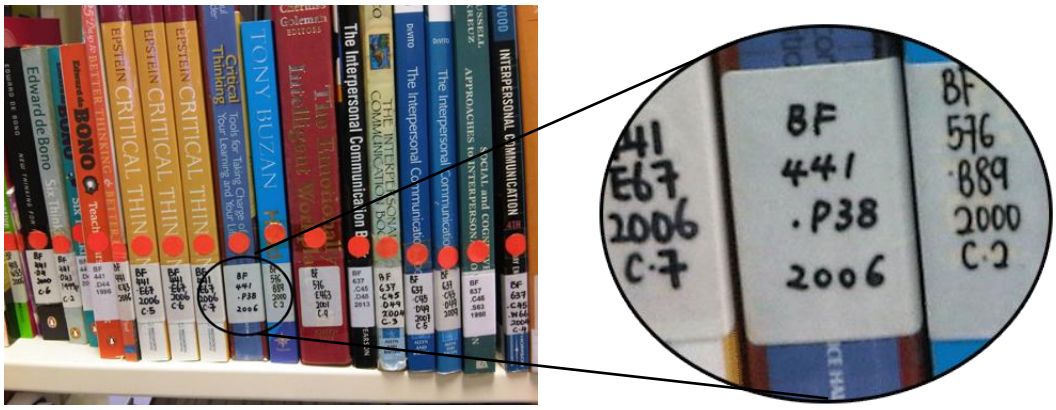


Figure 1.1.1: Books on the bookshelf and call numbers written on book label

If a book is misplaced, then its call number will be out-of-sequence with the call numbers of its neighbouring books. It is a routine for librarians to scan through the books the library shelves to look for misplaced literature. Due to the number of shelves found in a typical library, this is often a formidable task dreaded by the librarians. The task can, however, be assisted with a camera-equipped device which requires the librarian to simply take pictures of the books, the device will then automatically identify the misplaced books from the call numbers in the pictures taken. Such a device can be realized cheaply through a software application that can be installed on commodity smart phones which are ubiquitous nowadays. The software application can be naturally split into the following components:

1. A component to facilitate the picture-taking and picture-displaying.

2. A component to locate the labels in the books found within the pictures taken (segmentation),

3. A component for recognizing the call numbers printed on the labels (character recognition), and
4. A component for checking if the identified call numbers are in sequential order.
5. A component to highlight the locations of the misplaced books within the displayed pictures.

In this project, we are interested in the third of these components, that is, the problem of offline handwritten text recognition.

1.2 Project Scope and Objectives

As discussed in Section I, we plan to develop a mobile application which will enable librarians to search for misplaced books more efficiently. This mobile application is divided into several components, including image segmentation and handwriting recognition. The scope of my project is to produce the latter, that is, a handwriting recognizer for the mobile application.

The recognizer is to be able to identify the call numbers contained in images of book labels. Since call numbers consists of 26 capital letters and 10 digits from 0 to 9, the recognizer is expected to accept as input, images of any one of these 36 symbols. The recognizer produces as output a prediction of the character contained in each image.

The project will produce an implementation of the recognizer. This implementation is then tested against a dataset of test samples for performance and accuracy evaluation.

1.3 Methods/Technologies Involved

Programming Language

I choose to use the C programming language due to its flexibility and speed, which are often unmatched by higher level languages. Due to its low-level nature and small memory footprint, C codes often compiles into faster executable programs, compared to other higher-level languages.

Besides that, a very powerful library for image processing, Open Source Computer Vision Library (OpenCV), provides an easy-to-use programming interface in C, making C a natural choice for developing image processing applications.

Development Environment

Microsoft Visual Studio and MATrix LABoratory (Matlab) are used in this project. They are well-suited to my project as they respectively support development in C. Matlab is being used as it provides a large set of powerful predefined toolboxes such as Neural Pattern Recognition Tools, which can assist in my project when I want to test with different techniques.

Hardware used

A desktop computer is used in developing the handwriting recognizer, as well as in testing the recognizer's speed and accuracy.

CHAPTER 2: LITERATURE REVIEW

As far as I have searched, no known study has been performed on the problem of identifying library call numbers. On the other hand, many effective techniques have been developed for offline handwriting recognition. These techniques are based on several well-known methods in artificial intelligence, such as Artificial Neural Network (ANN). It is conceivable for these methods to be adapted for use in the present problem.

2.1 Common Methods Used in Character Recognition

2.1.1 Artificial Neural Network

An ANN consists of nodes which are placed on three regions: an input layer, a set of hidden layers, and an output layer. The nodes between subsequent layers are connected with weighted edges. Each node will calculate a value based on its input; the calculated value is then passed as input to subsequent nodes, after being adjusted with the weights on the edges which connects them.

In order to construct an ANN, a user needs only to specify the number of nodes and layers, how they are connected, and provide the ANN with sample sets of data with known output to train it. The training process adjusts the weights between the nodes in the ANN until a sufficiently close approximation of the output is achieved.

ANNs are widely used in handwriting recognition. (Nawrin & Hassan 2012) has reaches 99.95% in recognition Bangla Character by integrating the usage of freeman chain code and ANN. One advantage of using an ANN is that it is relatively simple to construct and hence saves time in their development. Matlab also provides a Neural Pattern Recognition Toolbox which allows user to use ANN easily.

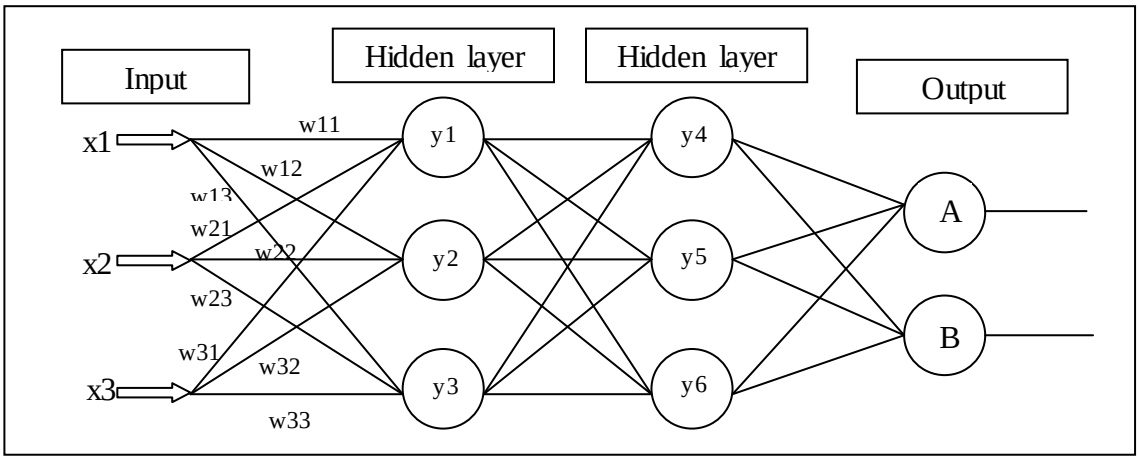


Figure 2.1.1.1: Implementation of Artificial Neural Network for handwriting recognition

2.1.2 Freeman Chain Code

As mentioned above, Nawrin & Hassan (Nawrin & Hassan 2012) reported that by using Freeman Chain Code as features, near 100% accuracy was achieved in classifying Bangla characters. Likewise, Gaurav & Jayashree (Gaurav & Jayashree 2013) mentioned the use of Freeman Chain Code as a feature for character recognition.

A Freeman Chain Code is an alternative representation (that is, a feature) of the image to be identified. It encodes the image as a sequence of numbers by travelling through the connection points of an image. Each move in a different direction is recorded with a number which ranges from 0 to 7. Each different character is expected to roughly translate into a specific chain code. Hence, the character of a chain code can be estimated by examining how well it matches chain codes of known characters.

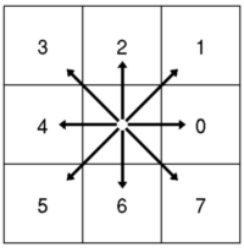


Figure 2.1.2.1: Direction representation by Freeman Chain Code

Example of chain code travels for character 'C': 3 4 4 4 4 5 5 6 6 6 6 6 6 6 7 7 0 0 0 0 1 1 2 3 5 5 4 4 2 2 2 2 1 0 0 7 0 1 3

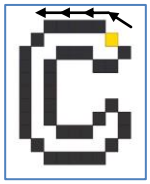


Figure 2.1.2.2: Freeman Chain Code travels for character 'C'

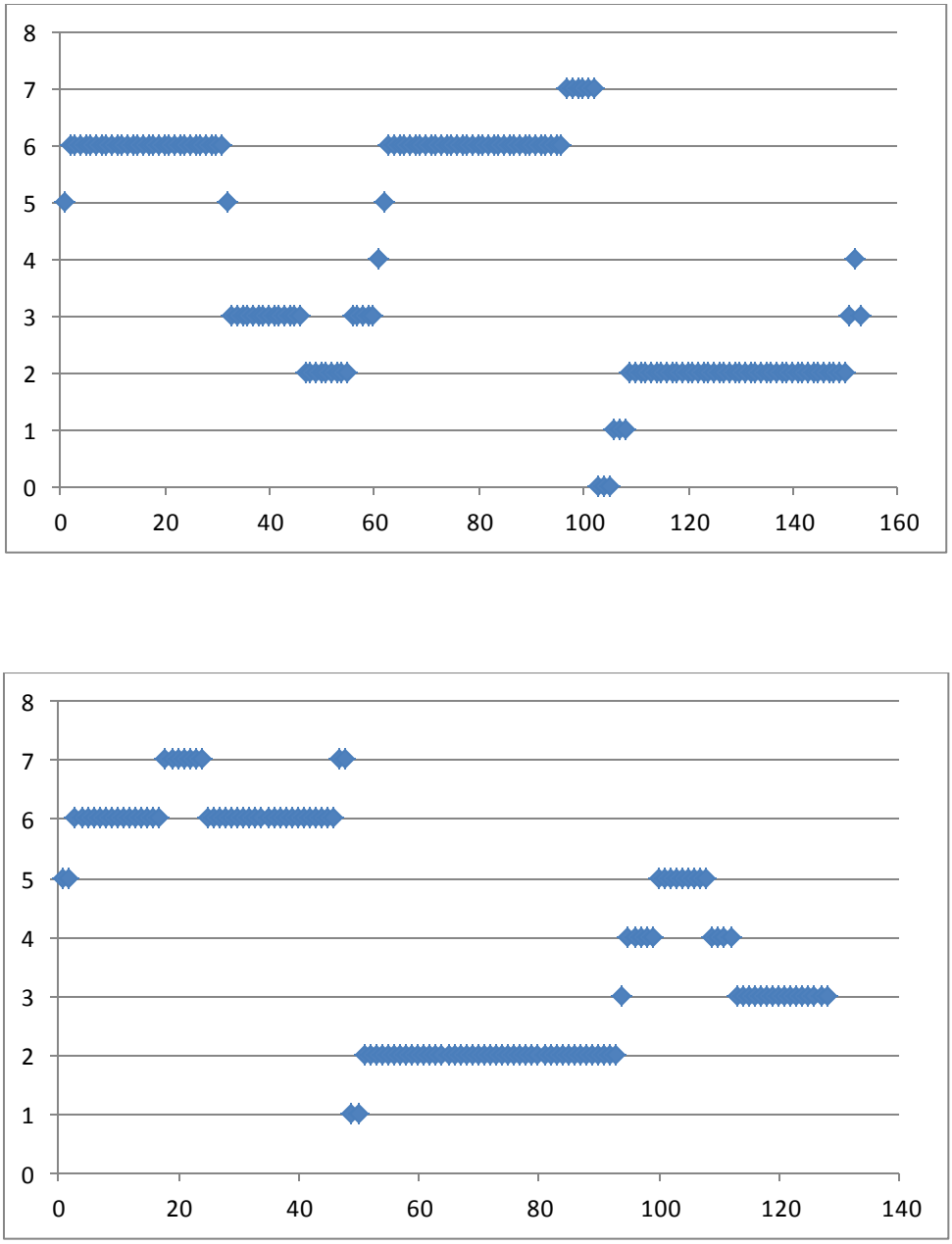


Figure 2.1.2.3: Example of chain code pattern shown for character 'V' and 'Y'

2.1.3 Edit Distance

One possible method to compare between two chain codes is through so-called Edit distance (or Levenshtein Distance), which measures the difference between two given sequences of characters. The edit distance between two sequences of character is, roughly speaking, the number of changes required to transform one sequence into the other. The distance obtained is zero if both sequences computed are exactly the same. The smaller the number of differences between two sequences, the higher is the possibility that they are alike.

Edit distances can be efficient computed through a dynamic programming algorithm. The use of edit distance in recognizing shapes was proposed in by Wu and Lai (Wu & Lai 2007). It is suited to this program as chain code can consider as a sequence of a characters.

In its most basic definition of the edit distance, the operations involved in the transformation of one sequence into the other are *addition*, *deletion* and *substitution*. The application of each operation in the transformation introduces a distance of 1 between the two sequences. This results in the following dynamic programming

$$\mathrm{ed}_{a,b}(i,j) = \begin{cases} \max(i,j) & , \min(i,j) = 0 \\ \min \begin{cases} \mathrm{ed}_{a,b}(i-1,j)+1 & , \text{deletion} \\ \mathrm{ed}_{a,b}(i,j-1)+1 & , \text{insertion} \\ \mathrm{ed}_{a,b}(i-1,j-1)+[a_i \neq b_j] & , \text{substitution} \end{cases} \end{cases}$$

CHAPTER 3: METHODOLOGY

In this project, I follow the development process stated in System Development Life Cycle (SDLC). SDLC divided into 5 phases which are Planning, Analysis, Design, Implementation, and Deliver.

1. Planning Phase

In this phase, the main objectives are defined and the reasons for building the system are clearly stated. The feasibility of the system was examined, and the following questions were answered.

Technical feasibility

- Can the system being build use current technology?
    - For our literature review, no.
- Can the recognizer be sufficiently accurate?
    - To be researched.

Organization feasibility

- Will the library staff be willing to use the system after it is complete?
    - Since no existing tools exist, this is very likely.
- Possible solutions if user resistances do happen.
    - We can provide the application for free.

Financial feasibility

- What is the price to obtain hardware, software tools required?
    - All the components required are free.

2. Analysis Phase

The requirements of the system were defined in this phase. A literature review was also performed, during which papers and journal publications are surveyed to identify possible solutions to the problem. Approaches that are most suited to the problem are identified.

3. Design Phase

There are some preparations required prior to the classification process. Sample images of character were extracted out and saved in a consistent naming format. A consistent naming format was essential in the testing stage, where a huge number of images are to be processed semi-automatically.

After summarizing the findings at previous phase, possible algorithms are then designed to solve the problem. Preliminary algorithms are then implemented. These algorithms are tested against the test samples, in order for me to examine the algorithms' strength and weaknesses, and to come up with solutions to overcome possible pitfalls. The insights gained are then used to design the final algorithm.

4. Implementation Phase

At this stage, the algorithm is implemented and its accuracy tested. This is the most critical part among the whole process as it affects the final results produced.

Several set of testing sample will be used to ensure the results gained are not biased. Also, testing is used to evaluate whether it is able to achieve the defined objectives and expected values during the Planning phase. Debugging will be performed to reduce the chances of possible errors.

5. Deliver Phase

At this stage, the final report is finalized with all details included before submission.

CHAPTER 4 ALGORITHM IMPLEMENTATION

4.1 Main Handwriting Recognizer Algorithm

Input of the program:

- Image of characters (.bmp)

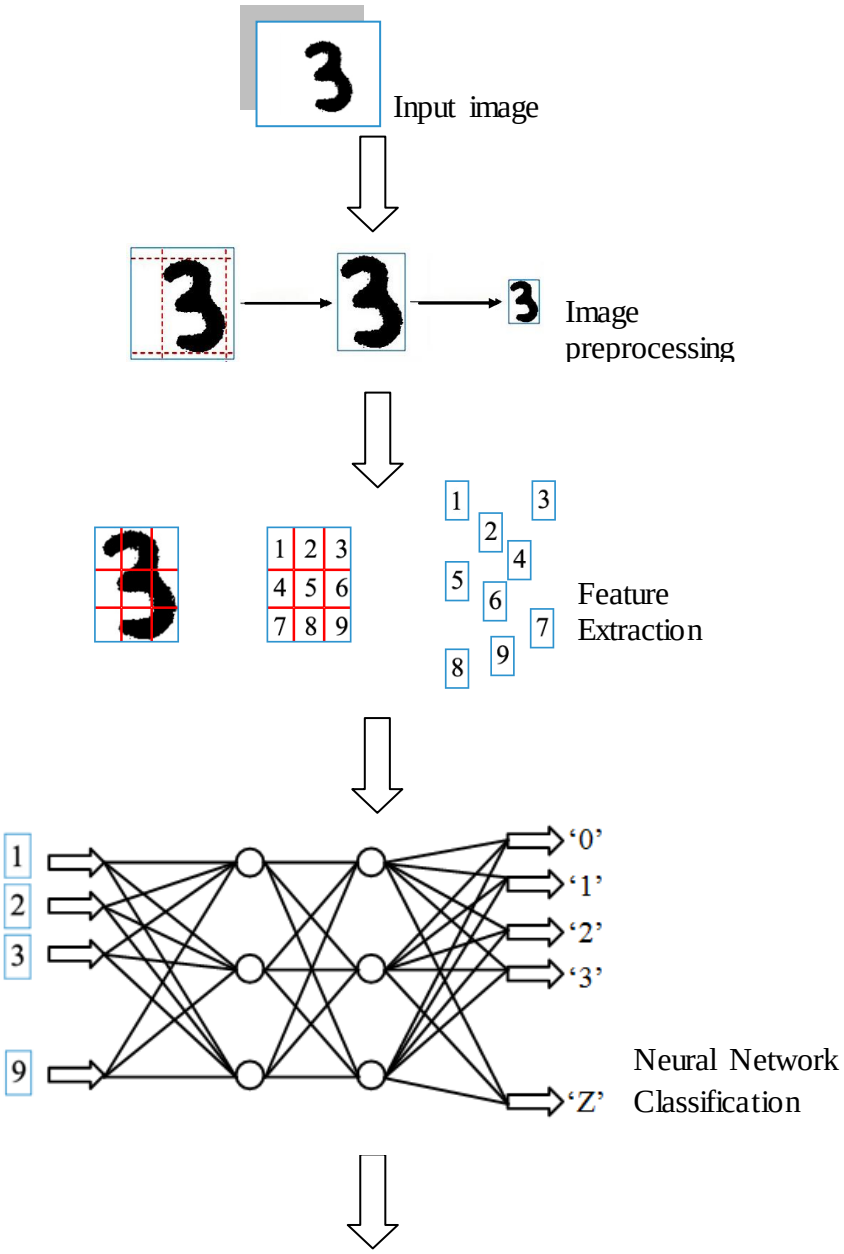
Output of the program:

- Character determined

The recognizer algorithm involves a three-stage identification process. In the first stage, the image of the character to be recognized is given to a neural network for identification. If the neural network is able to identify the character confidently, the identification stops. Otherwise, the image is sent for a second stage identification through an analysis of its Freeman chain code. If the character cannot be confidently identified through the chain code analysis, it is sent for a third stage identification which analyzes its chain code composition.

The entire procedure of the handwriting recognizer is summarized in Figure 4.1.2, where the details of each stage of the algorithm are given in Section 4.2.

Figure 4.1.1 gives a visual overview of the algorithm.


Input image
Image preprocessing
1 2 3
4 5 6
7 8 9
Feature Extraction
'0'
'1'
'2'
'3'
'Z'
Neural Network Classification

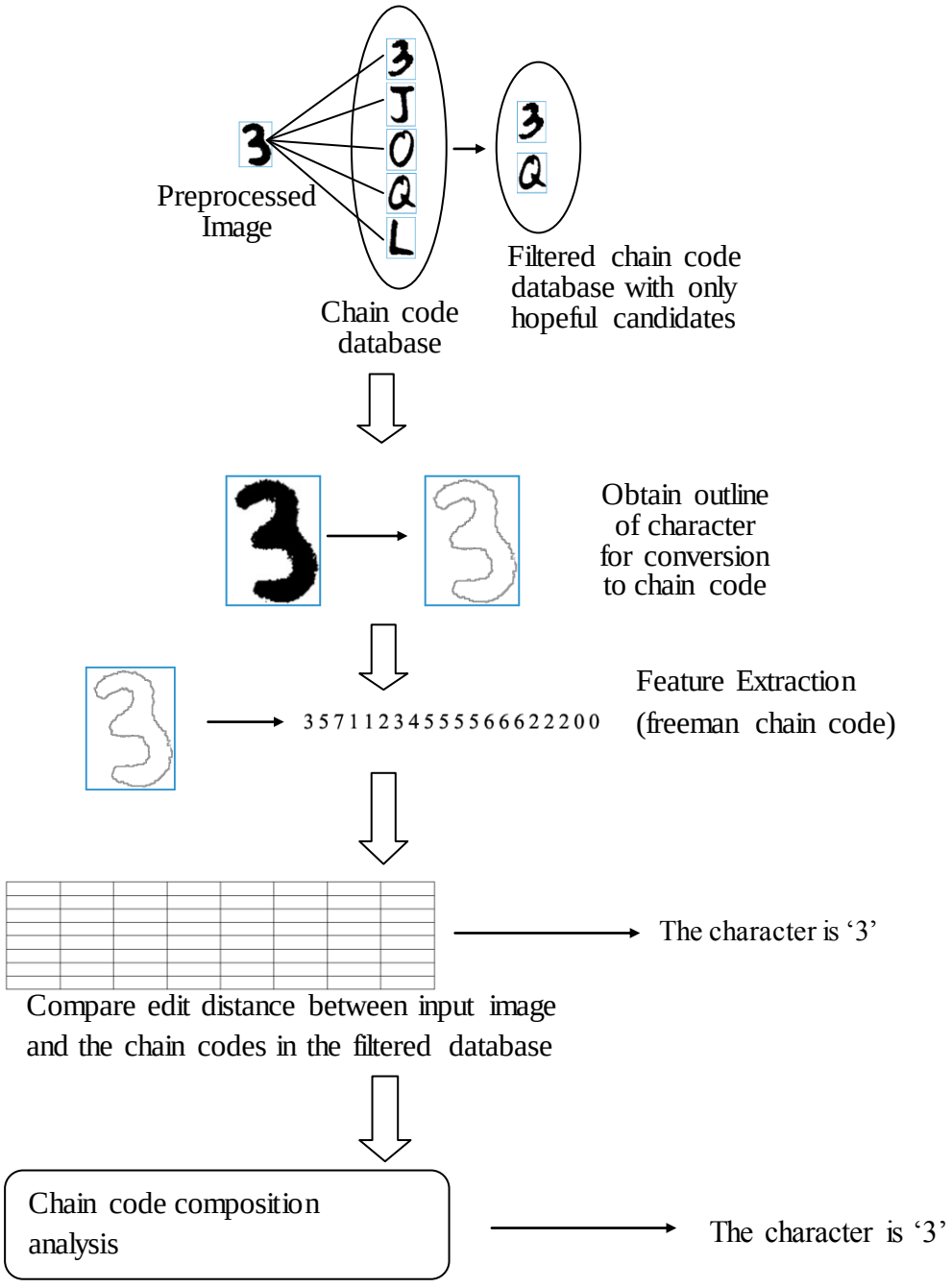


Figure 4.1.1: Overall system process

1. Read input image (.bmp).
2. Crop image.
3. Scale Image.
4. Break image into 3x3=9 partitions; get the total number of pixels in each partition.
5. Run data obtained in 4 into neural network trained.
    a. If neural network is able to identify the image with an output higher than 0.8, the character is determined and whole process is ended.
    b. Otherwise, the character is then passed to sample matching.
6. Filter the training sample by matching total number of pixels weight in each partition.
7. Find the outline of character.
8. Use freeman chain code to encode the image character.
9. Smooth chain code to reduce noise.
10. Identify character by matching with training sample using Edit Distance.
11. If character obtained is classified as 1, B, D, K or M, use chain code composition analysis and KNN to identify characters.

Figure 4.1.2: Main procedure of the program

4.2 Detailed Description of Main Procedure

4.2.1 Image pre-processing - Crop and Scale Image

Before the exact classification process takes place, pre-processing of image needs to be carried out. It will later help the data to be represented in a consistent form, as well as reduces the chances of inaccurate result.

This main purpose of this pre-processing is to standardize the size of the character. The size of handwritten characters often varies even when the same person is writing the same character. The resolution of the photo also results in different sizes of image. So, we need to crop the characters and scale it into similar sizes so that data representation does not show huge variant in length. For all the sample sets collected, I have cropped and scaled the image into a 30x50 pixels of size. Open Source Computer Vision (OpenCV) has been used to achieve this target.

OpenCV is an open source library which includes many image processing functions. It supports C/C++ language which can be integrated under Microsoft Visual Studio. OpenCV provides cvSetImageROI() and cvResize(), which can be used to crop and scale image. To crop an image, first we need to determine the margin of the four sides (top, bottom, left and right) of a single character.

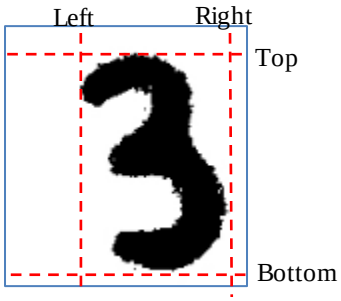


Figure 4.2.1.1: Image to be cropped

1. Read pixels line by line, from the top until the bottom of image.
    a. The first pixel read of the character will be the top margin.
2. Read pixels line by line, from the bottom until the top of image.
    a. The first pixel reads of the character will be the bottom margin.
3. Read pixels column by column, from left towards right of the image.
    a. The first pixel reads of the character will be the left margin.
4. Read pixels column by column, from right towards left of the image.
    a. The first pixel reads of the character will be the right margin.

Figure 4.2.1.2: Pseudocode for getting 4 side's boundary

After getting these margins, the image can be cropped and scaled by using cvSetImageROI() and cvResize().

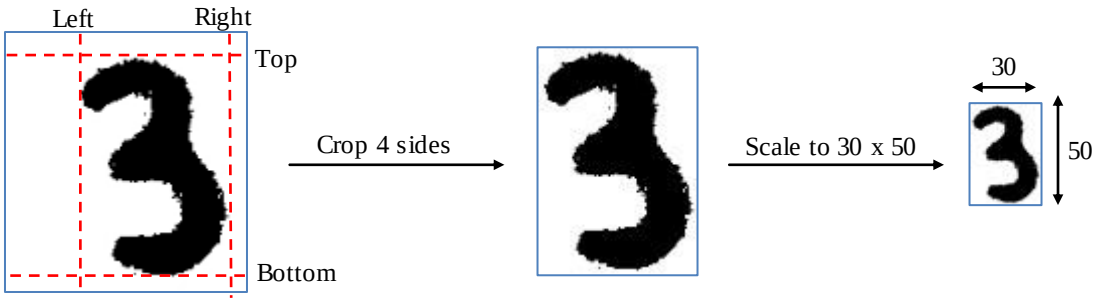


Figure 4.2.1.3: After crop and scale image

1. Calculate the new height and new weight of image by subtracting top, bottom, left, right margin space with the original image.
2. Use new height, new weight, left margin and top margin to crop the image by using cvSetImageROI().
3. Scale the image into 30x50 pixels by using cvResize().

Figure 4.2.1.4: Pseudocode for crop and scale image

4.2.2 Feature Extraction

For the classification through the neural network, I use the following as features. First, the image is broken into 3x3=9 partitions. For a greyscale image, each pixel assumes a value between 0 and 255. The summation of the pixel values in a partition is the weight of the partition. The vector which consists of the nine weights, each from a partition, is taken as the feature vector of an image.

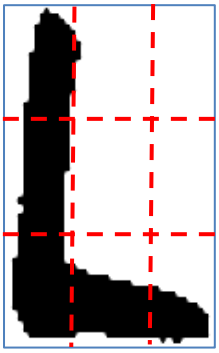


Figure 4.2.2.1: Image breaking into 3x3 partitions

4.2.3 Classification with Artificial Neural Network

The process next proceeds to character classification stage. A neural network is trained and will be used to recognize character (see Section 4.3.1 for details on the training of the neural network). The output of the neural network is an array with 36 elements that indicates the similarity respectively of the input image to the 36 symbols. Each element of the array ranges from 0.0 to 1.0. The first element of the array corresponds to the symbol '0' while the last element corresponds to the symbol 'Z'. The value of each array element indicates the similarity of the input image to its corresponding character. As an example, if the first element of the array (which corresponds to the symbol '0') gives a value of 0.2 and the second element of the array (which corresponds to the symbol '1') gives a value of 0.42, then the current input is more similar to '1' than '0'. Thus, by examining the values of the output array, an estimate of which character the input image corresponds to can be obtained.

Example of an output for an input image of the letter 'Z' (36 elements):

	1st	2nd	3rd	4th	5th	6th	…	32nd	33rd	34th	35th	36th
Character	'0'	'1'	'2'	'3'	'4'	'5'	…	'V'	'W'	'X'	'Y'	'Z'
Output	0.00	0.01	0.51	0.02	0.07	0.00		0.31	0.14	0.02	0.32	0.89

In the case above, the maximum value obtained the 36th element in the array, which has the value 0.89. Thus the character would be classified as 'Z'. However, not all inputs allow us to make decisions easily. For example consider the following:

	1st	2nd	3rd	4th	5th	6th	…	32nd	33rd	34th	35th	36th
Character	'0'	'1'	'2'	'3'	'4'	'5'	…	'V'	'W'	'X'	'Y'	'Z'
Output	0.00	0.01	0.41	0.02	0.07	0.00		0.38	0.24	0.02	0.32	0.29

	1st	2nd	3rd	4th	5th	6th	…	32nd	33rd	34th	35th	36th
Character	'0'	'1'	'2'	'3'	'4'	'5'	…	'V'	'W'	'X'	'Y'	'Z'
Output	0.00	0.01	0.00	0.02	0.07	0.00		0.08	0.01	0.00	0.10	0.00

In both of the cases above, the image matches several characters with similarly high scores, making it difficult to predict with confidence which character is it more likely to be. Furthermore, in these cases, the highest similarity score remains low, raising doubt on whether it could be case that the correct match (which should give a high similarity score) has been assigned a much lower value instead, due to some unforeseen reasons.

In order to avoid selecting a wrong character when the correct character has an unexpectedly low score, the character with the highest score from the neural network is selected only if its similarity value is at least 0.8. That is, whenever the maximum value of the neural network's output is equal or larger than 0.8, the character with the highest value is considered correct and the recognizer outputs this character. This threshold of 0.8 is chosen because it gives the highest classification accuracy on the training examples.

When the neural network gives no score of at least 0.8, an algorithm based on Freeman chain codes and edit distance will be invoked.

4.2.4 Image Preprocessing - Filtering Freeman Chain Code Database

In the second identification process, a database of Freeman chain codes is used to match the chain code of the input image (more details in Section 4.2.6-7). The database consists of seven sets of chain codes; each set consists of 36 chain codes for the 36 symbols. Initial tests show a comparison of all the chain codes (7x36=252 chain codes) in real-time is not feasible.

To overcome this problem, the chain code database is first filtered to remove the chain codes that are unlikely to match the chain code of the image. To do so, I compare the image of each chain code with the image of the input character. In order to facilitate this comparison,

I use the same features as that for the neural network (see Section 4.2.2). Each image gives a feature vector of 9 values, each indicating the total number of pixel values in a partition of the image. A simple method is used to compare between two feature vectors: the i^{th} element of a vector is compared to the i^{th} element of the other vector. Two vectors are considered unmatchable if:

- The value of any element in a vector is less than .2 of its corresponding element in the other vector.
- The value of any element in a vector is zero, but its corresponding element in the other vector has a value above 8000.

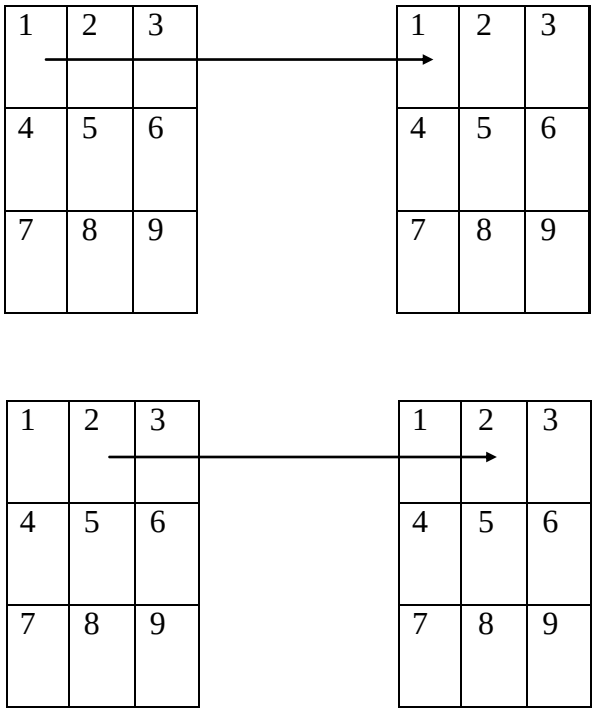


Figure 4.2.4.1: Comparing total number of pixel weight in partition 1 and 2

If the image for a chain code is deemed unmatchable to the input image, the chain code is removed from the database, and will not be used in the identification.

This method turned out to be very effective in filtering out the images of characters with dissimilar shape, since these shapes often differ greatly in at least one partition. For example, using this method, the characters 'J' and 'L' can be differentiated as they carry different pixel values in different partitions. The character 'L' has no black pixel in the partitions 3, 5 and 6, while the character 'J' has many black pixels in partitions 3 and 5.

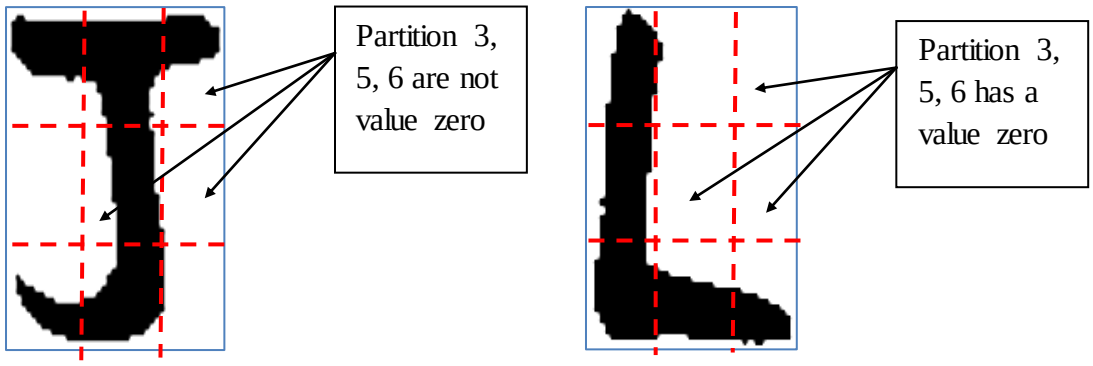


Figure 4.2.4.2: Filtering character 'J' and 'L'

Typically, this filtering results in the removal of 130~150 chain codes from the database.

4.2.5 Image Pre-processing – Obtain Outline of Character

The next step after filtering the chain code database is to transform the input image into a Freeman chain code. However, handwritten characters on library labels are typically written with marker pen, which produces very thick lines. Such lines cannot be easily converted into chain codes. To overcome this, I use the outline of the handwritten character to produce the chain code. This outline is obtained as follows: The image is scanned pixel-by-pixel, whenever a non-white pixel is encountered, its neighbouring 4 pixels are examined, if there exist any neighbouring pixel that is white, this pixel would be marked as an outline pixel. The pseudocode for this subroutine is in Figure 4.2.5.1.

1. Access image pixel-by-pixel
    a. If pixel is non-white (RGB value <= 128)
        i. Check its neightboring top, left, right, bottom pixels
            I. If any of the pixels is white (RGB > 128), the current pixel is marked as an outline pixel
            II. Otherwise, label the pixel as a non-output pixel
2. Color all non-outline pixels white.

Figure 4.2.5.1: Pseudocode for obtaining the character outline

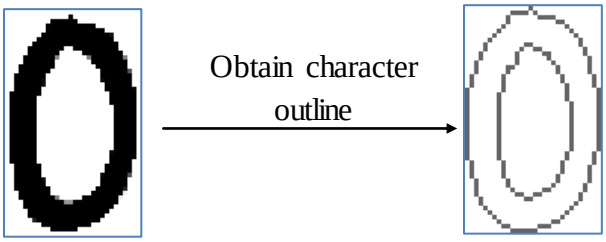


Figure 4.2.5.2: Image after obtaining the character's outline

4.2.6 Feature Extraction – Freeman Chain Code

The outline of the character obtained is to be converted into a Freeman chain code, in order to facilitate its comparison with the Freeman chain codes in the filtered database. This conversion is performed as follows: First, the algorithm first points at a start pixel chosen from the outline of the character. Then, a neighbouring pixel of the start pixel in the outline is chosen. The algorithm moves to this neighbouring pixel, *x* say, and register the corresponding chain code of the movement. After this, the algorithm proceeds to move to a neighbouring pixel of *x*, and register the corresponding movement (see Figure 2.1.2.1). Figure 4.2.6.1 demonstrates this procedure for the character 'C'; the movements registered for the three initial moves demonstrated is 3-4-4.

This traversal procedure repeats until the start pixel is encountered again, in which case, the entire outline has been traversed, and the sequence of movements registered is the chain code of the outline. (Some characters, such as 'O', produces two outlines. For such characters, only the outermost outline is used.)

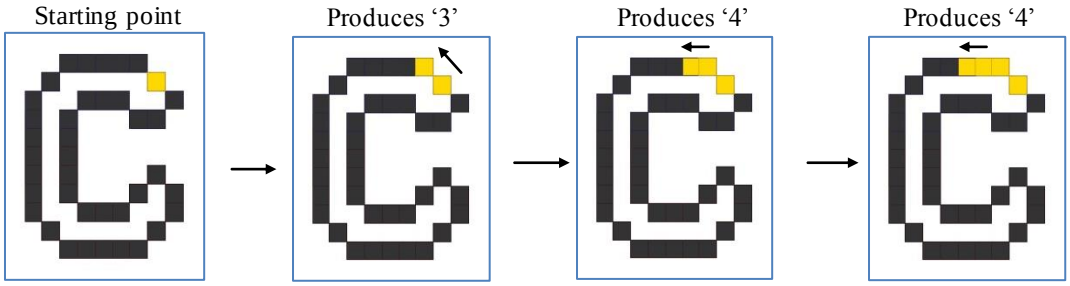


Figure 4.2.6.1: Example of chain code direction moving on character 'C'

This algorithm for converting an outline to a chain code, however, suffers a drawback: The choice of the starting point severely affects the chain code, and may result in vastly different chain codes even for the same character outline. To ensure that the same outline always result in the same chain code, a method which will deterministically choose the same pixel as the start pixel is required.

Some care is required when designing this method for choosing starting pixels. For example, consider the two possible outlines for the character 'W' below. If the topmost pixel is chosen as the starting pixel, then the two outlines will have different starting points, which will lead to very different chain codes. More precisely, the left-hand side outline which starts at the upper left corner will start with the codes 0 -> 7 -> 2 -> 0, while the right-hand side outline which starts at the upper right corner will start with the codes 7 -> 6 -> 3.

This example shows that the determination of the starting pixel based on examining the outline horizontally or vertically might not be useful as some characters will lead to different starting region depends on handwriting style.

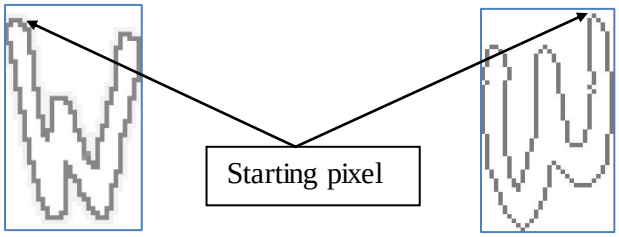


Figure 4.2.6.2: Detect starting point horizontally for character 'W'

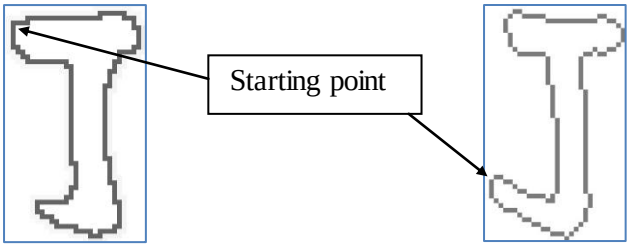


Figure 4.2.6.3: Detect starting point vertically for character 'W'

To solve this problem, the program read the points in a diagonal direction instead of horizontal or vertical direction; the first pixel detected would be the starting point of chain code. This can ensure that the chain code will start at the upper left region of every character.

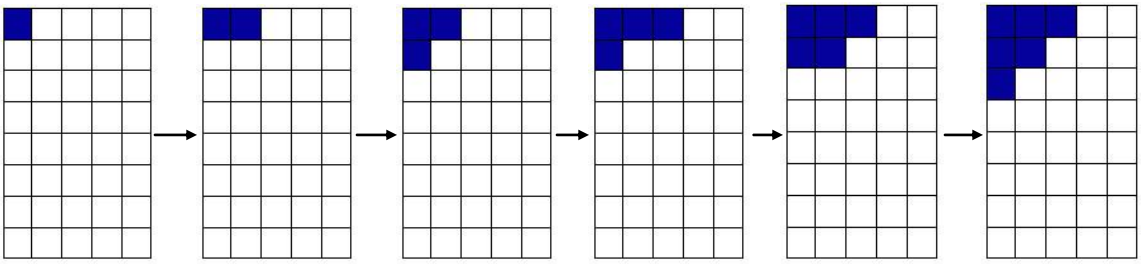


Figure 4.2.6.4: Detect starting point in diagonal direction

1. Initialize row to zero
2. Repeat the steps until points are detected
    a. Initialize i = 0, j = row
    b. Repeat the steps while i <= total number of rows and j > 0
        i. If the pixel at (i, j) is non-white, use it as the starting pixel
        ii. Otherwise, increase i by 1 and decrease j by 1
    c. Increase row by 1

Figure 4.2.6.5: Pseudocode use to detect starting point in diagonal direction

In order to generate the freeman chain code deterministically, the algorithm also needs to fix a method to generate the next move. This can be easily implemented by letting the algorithm try for possible movements in the following sequence: 0, 1, 2, 3, …, 7.

Finally, I show an example of a chain code obtained for the character 'X'.

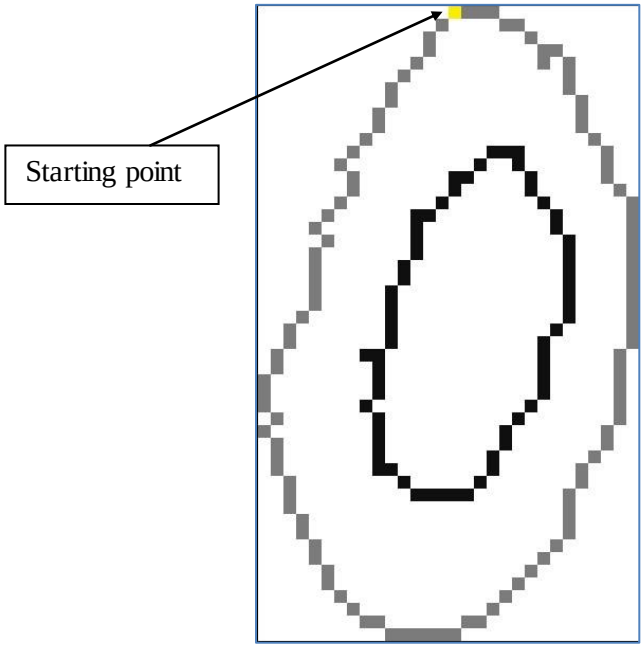


Figure 4.2.6.6: Image character 'O'

5 5 6 5 5 5 6 5 6 5 5 5 7 6 5 5 5 7 5 6 6 6 6 5 5 6 5 6 5 6 6 7 5 7 6 6 7 6 6 7
6 7 6 7 6 7 7 7 0 7 0 0 0 0 0 1 0 1 1 1 1 1 1 1 2 2 2 1 2 1 1 2 1 2 2 2 2 2 1 2
2 2 2 2 2 2 2 2 2 2 3 3 2 2 3 3 2 2 3 2 2 3 5 2 3 3 4 3 4 4 4

Figure 4.2.6.7: Chain code obtained for character 'O'

One problem with the chain codes generated this way is that they often contain noises. This is because character outlines are often not smooth due to limitations in image resolution. Such noises are immediately apparent when the chain code is examined carefully; it is likely to appear as an anomaly between a series of the same numbers. For example the first 5 codes for the character '0' in Figure 4.2.6.8 are 5-5-6-5-5. However, for a perfectly smooth outline of '0', a series of 5-5-5-5-5 should be expected. In this case, the character '6' is clearly a noise code which should be corrected to 5. By reducing noises a chain code can be smoothed. A chain code with less noise shows a clearer moving direction compared to chain code with noise.

To do this, I use an algorithm which examines all consecutive 5 elements of a chain code. The algorithm begins with the first 5 numbers of a chain code (i.e., the 1st, 2nd, 3rd, 4th, and 5th). The 3rd element is then compared with the 1st, 2nd, 4th, and 5th elements to determine if it is a noise, through the conditions stated in Figure 4.2.6.9, and altered accordingly if so. After this, the algorithm proceeds to examine the next 5 elements (2nd, 3rd, 4th, 5th, and 6th), comparing the 4th element with the 2nd, 3rd, 5th, and 6th. The process is repeated until the (n–2)-th element, where n is the total length of the chain code.

Original chain code obtained	5 5 6 5 5 5 6 5 6 5 5 5 7 6 5 5 5 7 5 6 6 6 6 5 5 6 5 6 5 6 6 7 5 7 6 6 7 6 6 7 6 7 6 7 6 7 7 7 0 7 0 0 0 0 0 1 0 1 1 1 1 1 1 1 2 2 2 1 2 1 1 2 1 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 3 3 2 2 3 3 2 2 3 2 2 3 5 2 3 3 4 3 4 4 4
Chain code obtained after reducing noise	5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 7 7 7 7 7 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2

	2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 4 4 4

Figure 4.2.6.8: Chain code gained before and after reducing noise

1. Copy original chain code into a new array
2. Repeat the step throughout the sequence of chain code by
    a. Copy i^{th} to i+5^{th} elements of chain code
    b. If any three of the 1^{st}, 2^{nd}, 4^{th} and 5^{th} elements are the same (c say)
        i. Set the 3^{rd} element to c
    c. Else if 4^{th} element is equal to 5^{th} element
        i. If 1^{st} element OR 2^{nd} element is equal to 5^{th} element
            I. Then 3^{rd} element = 5^{th} element
    d. Else if 1^{st} element is equal to 5^{th} element
        i. If 2^{nd} element equal to 1^{st} element + 1 OR 1^{st} element – 1
            I. Then 2^{nd} and 3^{rd} element is equal to 1^{st} element
        ii. If 4^{th} element equal to 1^{st} element + 1 OR 1^{st} element – 1
            I. Then 4^{th} and 3^{rd} element is equal to 1^{st} element
    e. Increase value of *i* by 1

Figure 4.2.6.9: Pseudocode used to reduce noise of chain code

4.2.7 Classification – Edit Distance

Finally, the converted chain code of the input character is compared to the chain codes in the filtered database. As mentioned, this comparison is performed using the edit distance. The original edit distance gives a flat penalty of 1 to each insertion or deletion. I use a modified edit distance which penalizes less on the character changes that are considered less severe. For example, a move to the left is not very different from a move to the upper left, compared to the difference between an upward move and a downward move. The modified edit distance divides the level of similarities into 5 and uses a range of 0.0–2.4 as penalties. A difference which is considered less severe introduces less into the edit distance, while a difference which is more severe introduces more distance. For example, the distance introduced when matching different codes to '0' (⟶) is listed below.

Chain code of input	Chain code of sample in database	Difference
(0) ⟶	(0) ⟶	0.0
(0) ⟶	(1) ↗	0.2
(0) ⟶	(2) ↑	1.5
(0) ⟶	(3) ↖	2.0
(0) ⟶	(4) ⟵	2.4
(0) ⟶	(5) ↙	2.0
(0) ⟶	(6) ↓	1.5
(0) ⟶	(7) ↘	0.2

Figure 4.2.7.1: Example of matching different codes to '0'

The dynamic programming for computing this modified edit distance is given in Figure 4.2.7.2.

1. Store chain code of training sample in arrayT
2. Store chain code of testing sample in arrayS
3. Initialize first row of 2D matrix as 0.0
4. Initialize first column of 2D matrix as 0.0
5. For each element of arrayS,
    a. For each element of arrayT
        i. If element of arrayT – arrayS = 0
            I. Then difference = 0.0
        ii. If element of arrayT – arrayS = 1 OR 7
            I. Then difference = 0.2
        iii. If element of arrayT – arrayS = 2 OR 6
            I. Then difference = 1.5
        iv. If element of arrayT – arrayS = 3 OR 5
            I. Then difference = 2.0
        v. If element of arrayT – arrayS = 4
            I. Then difference = 2.4
        vi. Compute the minimal value by comparing matrix[x-1][y] + 1 (deletion), matrix[x][y-1] + 1 (insertion), matrix[x-1][y-1] + num (substitution).

Figure 4.2.7.2: Pseudocode use for calculate edit distance

Character recognition with solely chain codes achieves fairly high accuracy. When tested against 3 set of test samples (i.e. 3x36=108 characters), 105 of the characters were correctly identified. The characters which the chain code comparison failed to identify are for the characters ‘1’, ‘B’, ‘D’, ‘K’ and ‘M’.

4.2.8 Classification – Chain Code Composition Analysis

As mentioned, chain codes comparison failed to identify the characters '1', 'B', 'D', 'K' and 'M'. A better method to analyze these characters is by examining the composition of their chain codes. The composition of a chain code is the percentage of '0'. '1', …, '7' it contains. From my analysis, I found the letters '1', 'B', 'D', 'K' and 'M' to be particularly different in their chain code compositions. Hence, the differences in their chain code composition can be used effectively to distinguish them from the other characters. To compute the difference between two chain code compositions, the '0', '1', …, '7' composition of the chain codes respectively are first computed. Then, the difference between their individual '0', '1', …, '7' composition is computed. Finally, the difference between the two chain code compositions is the sum of all the differences between the individual '0', '1', …, '7' compositions. This is shown in Figure 4.2.8.2.

Code	Composition of				Differences against		
	Chain Code 1	Chain Code 2	Chain Code 3	Input Chain Code	Chain Code 1	Chain Code 2	Chain Code 3
0	10	12	3	8	2	4	5
1	32	10	23	12	20	2	11
2	33	32	35	30	3	2	5
3	52	17	27	24	28	7	3
4	22	21	29	15	7	6	14
5	11	33	17	33	22	0	16
6	2	8	6	7	5	1	1
7	30	2	4	28	2	2	24
				∑ =	89	24	79

Figure 4.2.8.1: Example of calculating difference between the input chain code with three chain codes from the database

1. Sequence of chain code of training sample is separated into groups according to its number (0-7), for each group, the number of members are counted and store in an arrayT.
2. Same as testing sample, the summation of chain code is stored in arrayS.
3. Substract each element of arrayS and arrayT. The value of substraction would be accumulated and the total difference calculated.

Figure 4.2.8.2: Pseudocode use for calculate total difference

While the chain code composition effectively distinguishes the chain codes that are different, it is not consistent in giving the highest similarity scores to the correct chain code. To overcome this, a KNN strategy is adopted. That is, K chain codes in the database with the closest compositions to the input chain code are obtained. Then, the character with the most chain codes among the K ones are given as the output character. In my test, I found that letting K=3 achieves the highest accuracy possible. That is, if any two chain codes in the nearest 3 found are for the same character, then the classifying process is completed and the character is given as output. A problem, however, arises when all 3 nearest characters are difference.

In such a situation, I use the KNN results with the output values of the ANN (explained in 4.1.4) to modify the difference score from the chain code composition. This new difference is computed as (0.1 * original difference for character)*(1−ANN score for character). Then, the character with the smallest difference is selected as the output.

When tested, this new score produces favourable accuracy in identifying the characters '1', 'B', 'D', 'K' and 'M'.

1. Find 3 chain codes with the closest chain code composition with the input chain code
2. For each value, calculate its new difference by using equation stated above.
3. The character with the smallest new difference is given as the output.

Figure 4.2.8.3: Pseudocode for chain code composition with KNN

4.2.9 Overall Results Obtained

The overall three-stage system (ANN, Chain code-edit distance analysis, Chain code composition with KNN) is tested against three sets of test samples. That is, a total of 3*36=108 characters. The following shows the result:

Character	Set 1	Set 2	Set 3	Correctness
0	0	0	0	3/3
1	1	1	1	3/3
2	2	2	2	3/3
3	3	3	3	3/3
4	4	4	4	3/3
5	5	5	5	3/3
6	6	6	6	3/3
7	7	7	7	3/3
8	8	8	8	3/3
9	9	9	9	3/3
A	A	A	A	3/3
B	B	B	B	3/3
C	C	C	C	3/3
D	D	D	D	3/3
E	E	E	E	3/3
F	F	F	F	3/3
G	G	G	G	3/3
H	H	H	H	3/3
I	I	I	I	3/3
J	J	J	J	3/3
K	K	K	K	3/3
L	L	L	L	3/3
M	M	M	M	3/3
N	N	N	N	3/3
O	O	O	O	3/3
P	P	P	P	3/3

Q	Q	Q	Q	3/3
R	R	R	R	3/3
S	S	S	S	3/3
T	T	T	T	3/3
U	U	U	U	3/3
V	V	V	V	3/3
W	W	W	W	3/3
X	X	X	X	3/3
Y	Y	Y	Y	3/3
Z	Z	Z	Z	3/3

Figure 4.2.9.1: Results obtained

Testing Sample Set = 3

Number of character tested = 3 x 36 = 108

Number of character failed = 0

Number of character mismatched = 0

Accuracy = 108/108 = 100%

Performance-wise, the three-stage recognizer is able to analyze each input character nearly instantaneously.

4.2 Sample Preparation

The test data and training comes from a pool of 60 sets of characters, cropped out mostly from the labels on the books of 50~60 photos taken from the UTAR library.

50 sets (50*36=1800 characters) of the data are used to train the ANN. 7 sets (7*36=252) of the data are used to generate the Freeman chain code database used in the chain code edit distance analysis as well as the chain code composition analysis. The remaining 3 sets of characters are used as test samples to determine the accuracy of the final system, as well as for preliminary tests.

4.3 Training and Testing Engine

4.3.1 Artificial Neural Network

Training Stage

For the artificial neural network, the Matlab library is used, since it provides a comprehensive pattern recognition tool set that makes the use of its neural network convenient. As mentioned, 50 sets of training samples are used in training the neural network. The feature extraction method discussed in 4.1.2 was implemented; the resultant data was stored in a text file. The text file is then loaded together with the training target in order to train the network. Results produced from neural network varies each time the training is completed, thus there is a need to save the network each time the training completes. The neural network with the highest accuracy tested against the test samples is selected for my system.

1. Read training input
2. Read training target
3. Define layer of network as 7
4. Create the neural network by using patternnet()
5. Setup division of data,
    a. Training = 100%
    b. validation = 0%
    c. testing = 0%
6. Train and save the network

Figure 4.3.1.1: Pseudocode use for training neural network

Testing Stage

After a network completed its training stage, testing is carried out to determine its accuracy. Test samples are loaded into the network and the results obtained. The output of the neural network is stored in an array which holds 36 rows. Each row holds a score which is indicative of the input's similarity to the character represented by the row. For example, the score in the first row indicates the similarity of the test sample to the character '0', the score in the second indicates the similarity of the test sample to the character '1', etc.

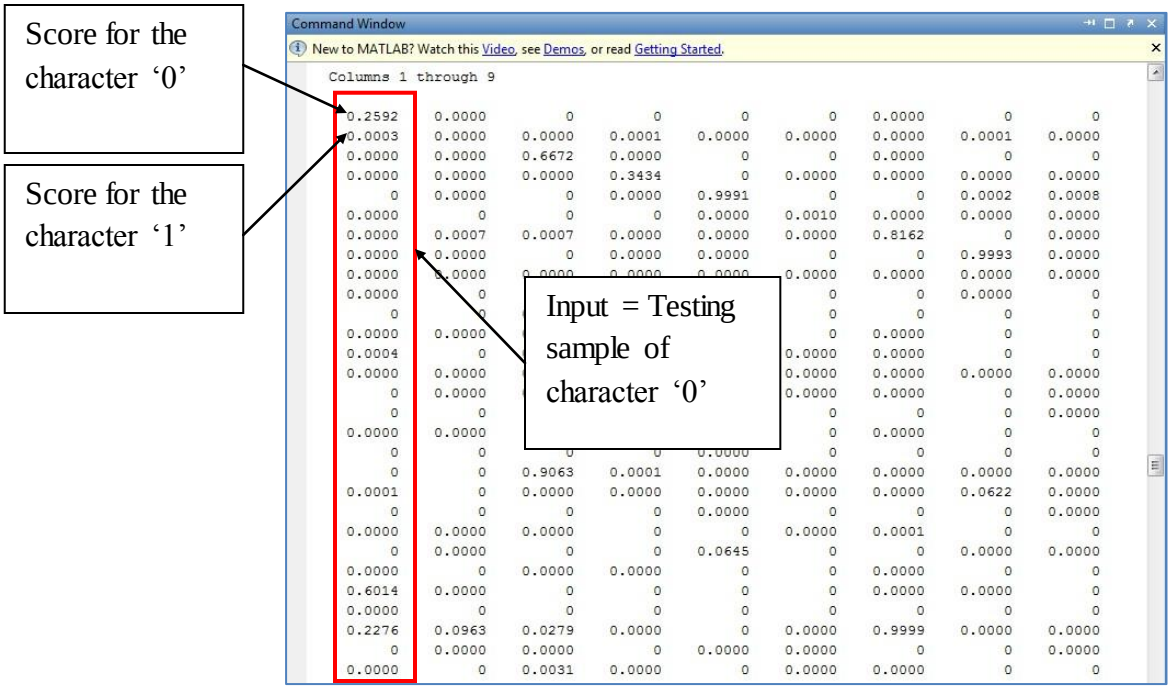


Figure 4.3.1.2: Sample output of neural network

1. Load trained network
2. Load testing sample
3. Input testing sample into neural network and gained results

Figure 4.3.1.3: Pseudocode used for testing stage

While carrying out training I noticed that whenever the score is 0.8 or above, the character identified would always be correct. Hence 0.8 is taken as a threshold in my algorithm. The ideal trained network will be the one with more result produced with probability 0.8 and above. Thus, training and testing is carried out numerous times and I chose the neural network with an output that is nearest to the ideal condition.

4.3.2 Freeman Chain Code Recognizer

For the Freeman chain code analysis, no training to fit parameters is required, other than the pre-processing required to prepare the chain code database.

CHAPTER 5: DISCUSSION

5.1 Achievement, Future Improvement and Conclusion

The project succeeded in its main objective, which is to achieve near-perfect accuracy in recognizing handwritten text on library labels. Tests conducted on 3 test sets of 3*36=108 characters showed 100% accuracy. Performance-wise, the recognizer is able to analyze each image instantaneously. This implies that the resultant system is a suitable candidate for use in our target mobile application for detecting misplaced books on library shelfs.

As the neural network used in the project is from Matlab, a more efficient implementation in C is required.

In conclusion, the project has achieved its objectives. The problems encountered have been solved by taking the suggestion by my advisor. I learned a lot from working on this project, and I am satisfied with seeing an algorithm that I designed works.

BIBLIOGRAPHY

Nawrin Binte Nawab, M. N. Hassan, 2012, 'Optical Bangla Character Recognition using Chain Code', *IEEE/OSA/IAPR International Conference on Informatics, Electronics & Vision* 2012.

Gaurav Y.Tawde, Mrs. Jayashree M.Kundargi, 2013, 'an Overview of Feature Extraction Techniques in OCR for Indian Scripts Focused on Offline Handwriting', *International Journal of Engineering Research and Applications (IJERA)*, Vol.3, Issue 1, January – February 2013, pp.919-926.

W-Y Wu, C-S Lai, 2007, 'Shape Recognition using fuzzy string- matching technique', *The Imaging Science Journal*, Vol. 5, pp.223-231.

APPENDIX – A COMPLETE CODING OF THE WHOLE PROGRAM

Appendix A - Program Used for Crop and Scale Image

```
#include "opencv2\core\core.hpp"
#include "opencv2\highgui\highgui.hpp"
#include "opencv\cv.h"
#include "opencv\highgui.h"
#include <iostream>
#include <string.h>
using namespace std;
using namespace cv;

int main(int argc, char *argv[])
{
	char character[36] =
{'0','1','2','3','4','5','6','7','8','9','A','B','C','D','E','F','G','H','I'
,'J','K','L','M','N','O','P','Q','R','S','T','U','V','W','X','Y','Z'};
	char setK[10] = {'0','1','2','3','4','5','6','7','8','9'};
	char setL[6] = {'1','2','3','4','5','6'};
	int imgResult;
	char imgName[10] = {};
	char inputName[11] = {};
	int bottom, top, left, right;
	IplImage* img = 0;
	int height,width,step,channels, depth, newHeight, newWidth;
	uchar *data;
	int i,j;

	for(int l = 0; l < 5; l++){
		for(int k = 0; k < 10; k++){
			for(int sample = 0; sample < 36; sample++){
				top = 0;
				bottom =0;
				left = 0;
				right = 0;
				imgName[0] = character[sample];
				imgName[1] = ' ';
				imgName[2] = '(';
				imgName[3] = setL[l];
```

```
                        imgName[4] = setK[k];
                        imgName[5] = ')';
                        imgName[6] = '.';
                        imgName[7] = 'b';
                        imgName[8] = 'm';
                        imgName[9] = 'p';
                        imgResult = sprintf(inputName,
"%c%c%c%c%c%c%c%c%c%c", imgName[0], imgName[1], imgName[2], imgName[3],
imgName[4], imgName[5], imgName[6], imgName[7], imgName[8], imgName[9]);
                        img=cvLoadImage(inputName,
                  CV_LOAD_IMAGE_GRAYSCALE);
                        if(!img) {
                              printf("Could not load image file: %s\n",
inputName);
                              system("PAUSE");
                              return -1;
                        }
                        depth    = img->depth;
                        height     = img->height;
                        width      = img->width;
                        step       = img->widthStep;
                        channels   = img->nChannels;
                        data       = (uchar *)img->imageData;

                        //get top margin
                        i = 0;
                        while(top == 0 && i < height){
                              j = 0;
                              while(top == 0 && j<width){
                                    if(data[i*step+j] <= 128)
                                          top = i*step+j;
                                    j++;
                              }
                              i++;
                        }
                        //get bottom margin
                        i = (height*step)-(step-width+1);
                        while(bottom == 0 && i > 0){
                              if(data[i] <= 128)
                                    bottom = i;
                              i--;
                        }
```

```
                        //get left side margin
                        j = 0;
                        while(left == 0 && j < width){
                                i = 0;
                                while(left == 0 && i < height){
                                        if(data[i*step+j] <= 128)
                                                left = i*step+j;
                                        i++;
                                }
                                j++;
                        }
                        //get right side margin
                        j = width - (step - width + 1);
                        while(right == 0 && j < width){
                                i = 0;
                                while(right == 0 && i < height){
                                        if(data[i*step+j] <= 128)
                                                right = i*step+j;
                                        i++;
                                }
                                j--;
                        }

                        //crop image
                        //new height and width after cropped
                        newHeight = (height - (((top - step)/step) +
height - (((bottom + step)/step)))) + 4;
                        newWidth = (width - (((left - 1)%step) + width -
(((right + 1)%step)))) + 4;
                        cvSetImageROI(img, cvRect(((left - 1)%step)-2,
((top - step)/step)-2, newWidth, newHeight));

                        //resize image
                        CvSize size = cvSize(30,50);
                        IplImage* tmpsize=cvCreateImage(size,8,0);
                        cvResize(img,tmpsize,CV_INTER_LINEAR);

                        cvSaveImage(inputName,tmpsize);
                        cvShowImage("mainWin", tmpsize);
                }
        }
    }
    return 0;
}
```

Appendix B - Program Used to Train Neural Network

```
% read input
inputs = train_input';
targets = train_target';

% Create a Pattern Recognition Network
hiddenLayerSize = 7;
net = patternnet(hiddenLayerSize);

% Setup Division of Data for Training, Validation, Testing
net.divideParam.trainRatio = 100/100;
net.divideParam.valRatio = 0/100;
net.divideParam.testRatio = 0/100;

% Train and save the Network
[net,tr] = train(net,inputs,targets);
save('D:\\net.mat', 'net')
```

Appendix C - Program Used to Run Neural Network Character Classification

```
% read input - feature extraction data
testingInput8 = test8';
testingInput9 = test9';
testingInput10 = test10';

% load network trained
load('net.mat');

% Test input with Network trained/load
testingOutput8 = net(testingInput8);
testingOutput9 = net(testingInput9);
testingOutput10 = net(testingInput10);

% Output results to txt file
fid = fopen('D:\\result58.txt','w');
fprintf(fid,'%.4f \n',testingOutput8);
fclose(fid);
fid2 = fopen('D:\\result59.txt','w');
fprintf(fid2,'%.4f \n',testingOutput9);
fclose(fid2);
fid3 = fopen('D:\\result60.txt','w');
fprintf(fid3,'%.4f \n',testingOutput10);
fclose(fid3);
```

Appendix D - Program Used for Edit Distance Classification and Chain Code Composition Analysis

```
#include "opencv2\imgproc\imgproc.hpp"
#include "opencv2\core\core.hpp"
#include "opencv2\highgui\highgui.hpp"
#include "opencv\cv.h"
#include "opencv\highgui.h"
#include <iostream>
#include <string.h>
#include <stdlib.h>
#include <stdio.h>
#define MIN3(a, b, c) ((a) < (b) ? ((a) < (c) ? (a) : (c)) : ((b) < (c) ?
(b) : (c)))
using namespace std;
using namespace cv;

int main(int argc, char *argv[])
{
	char target;
	int c_sample=99;
	while(c_sample!=100)
	{
		printf("================================\n");
		printf("Enter target = ");//read testing sample. E.g. A
		scanf("%c", &target);
		fflush(stdin);
		printf("Enter sample = ");//read its position. E.g. Position
for 'A' is 10
		scanf("%d", &c_sample);
		fflush(stdin);

		char character[36] =
{'0','1','2','3','4','5','6','7','8','9','A','B','C','D','E','F','G','H','I',
'J','K','L','M','N','O','P','Q','R','S','T','U','V','W','X','Y','Z'};
		char set[7] = {'1','2','3','4','5','6','7'};
		int library[8][288] = {0};
		int library2[300][250];
		int imgResult;
		char imgName[10] = {};
		char inputName[11] = {};
```

```
		IplImage* img = 0;
		IplImage* img2 = 0;
		IplImage* img3 = 0;
		IplImage* img4 = 0;
		uchar *data;
		uchar *data2;
		uchar *data3;
		uchar *data4;
		int height,width,step,channels, depth;
		int i,j;
		int startI, startJ, partition;
		int filter;
		int filterCha[250] = {};
		int numChaMatch = 0;
		int partitionT[9] = {};
		double ANN[36][36];

		//read testing sample image
		imgName[0] = target;
		imgName[1] = ' ';
		imgName[2] = '(';
		imgName[3] = '5';
		imgName[4] = '8';
		imgName[5] = ')';
		imgName[6] = '.';
		imgName[7] = 'b';
		imgName[8] = 'm';
		imgName[9] = 'p';
		imgResult = sprintf(inputName, "%c%c%c%c%c%c%c%c%c%c",
imgName[0], imgName[1], imgName[2], imgName[3], imgName[4], imgName[5],
imgName[6], imgName[7], imgName[8], imgName[9]);

		img3=cvLoadImage(inputName, CV_LOAD_IMAGE_GRAYSCALE);
		if(!img3) {
			printf("Could not load image file: %s\n", inputName);
			system("PAUSE");
			return -1;
		}
		depth    = img3->depth;
		height    = img3->height;
		width     = img3->width;
		step      = img3->widthStep;
```

```
            channels  = img3->nChannels;
            data3      = (uchar *)img3->imageData;
            printf("Testing image: %c%c%c%c%c%c%c%c%c%c\n", imgName[0],
imgName[1], imgName[2], imgName[3], imgName[4], imgName[5], imgName[6],
imgName[7], imgName[8], imgName[9]);
            //read ANN results
            FILE *ptr;
            if(imgName[4] == '8')
                  ptr = fopen("result58.txt", "r");
            else if(imgName[4] == '9')
                  ptr = fopen("result59.txt", "r");
            else
                  ptr = fopen("result60.txt", "r");

            for(int i = 0; i < 36; i++){
                  for(int j = 0;j < 36; j++){
                        fscanf(ptr, "%lf ",&ANN[i][j]);
                  }
                  fscanf(ptr,"\n");
            }
            fclose(ptr);
            //find maximum value
            double maxANN = 0;
            int ch;
            for(int i = 0; i < 36; i++){
                  if(ANN[c_sample][i] > maxANN){
                        maxANN = ANN[c_sample][i];
                        ch = i;
                  }
            }
            if(maxANN >= 0.8){
                  printf("ANN: The character is predicted to be %c\n",
character[c_sample]);
            }
            else{
                  //filtering - break testing img into 3x3 partition
                  partition = 1;
                  while(partition < 10){
                        startI = 0;
                        startJ = 0;
```

```
                        if(partition == 1 || partition == 4 || partition == 7)
                                startJ = 0;
                        if(partition == 2 || partition == 5 || partition == 8)
                                startJ = 10;
                        if(partition == 3 || partition == 6 || partition == 9)
                                startJ = 20;
                        if(partition == 1 || partition == 2 || partition == 3)
                                startI = 0;
                        if(partition == 4 || partition == 5 || partition == 6)
                                startI = 17;
                        if(partition == 7 || partition == 8 || partition == 9)
                                startI = 34;

                        for(i=startI;i<startI+17;i++)
                                for(j=startJ;j<startJ+10;j++)
                                        if(data3[i*step+j] != 255)
                                                partitionT[partition - 1] += (255-
data3[i*step+j]);
                        partition++;
                }

                //read training sample
                for(int k = 0; k < 7; k++){
                        for(int sample = 0; sample < 36; sample++){
                                int partitionS[9] = {};
                                imgName[0] = character[sample];
                                imgName[1] = ' ';
                                imgName[2] = '(';
                                imgName[3] = '5';
                                imgName[4] = set[k];
                                imgName[5] = ')';
                                imgName[6] = '.';
                                imgName[7] = 'b';
                                imgName[8] = 'm';
                                imgName[9] = 'p';
                                imgResult = sprintf(inputName,
"%c%c%c%c%c%c%c%c%c%c", imgName[0], imgName[1], imgName[2], imgName[3],
imgName[4], imgName[5], imgName[6], imgName[7], imgName[8], imgName[9]);

                                img4=cvLoadImage(inputName,
CV_LOAD_IMAGE_GRAYSCALE);
```

```
                                    partition = 1;
                                    filter = 0;
                                    while(partition < 10){
                                          startI = 0;
                                          startJ = 0;
                                          if(partition == 1 || partition == 4 ||
partition == 7)
                                                startJ = 0;
                                          if(partition == 2 || partition == 5 ||
partition == 8)
                                                startJ = 10;
                                          if(partition == 3 || partition == 6 ||
partition == 9)
                                                startJ = 20;
                                          if(partition == 1 || partition == 2 ||
partition == 3)
                                                startI = 0;
                                          if(partition == 4 || partition == 5 ||
partition == 6)
                                                startI = 17;
                                          if(partition == 7 || partition == 8 ||
partition == 9)
                                                startI = 34;

                                          for(i=startI;i<startI+17;i++)
                                                for(j=startJ;j<startJ+10;j++)
                                                      if(data4[i*step+j] != 255)

      partitionS[partition - 1] += (255-data4[i*step+j]);

                                          if(partitionS[partition - 1] <
(partitionT[partition - 1]*0.2))
                                                filter++;
                                          else if(partitionT[partition - 1] == 0)
                                                if(partitionS[partition - 1] >
8000)
                                                      filter++;
                                          else if(partitionS[partition - 1] == 0)
                                                if(partitionT[partition - 1] >
8000)
                                                      filter++;

                                          partition++;
                                    }
```

```
                                    if(filter < 2){
                                          filterCha[numChaMatch] =
(36*k)+sample;
                                          numChaMatch++;
                                    }
                              }
                        }

                        for(int k = 0; k < numChaMatch+1; k++){
                              char imgName[7] = {};
                              char inputName[8] = {};
                              int chainCode[30000] = { };
                              int sequence = 0;
                              int startWidth = 0;
                              int startHeight = 0;
                              int start = 1;

                              imgName[0] = character[filterCha[k]%36];
                              imgName[1] = ' ';
                              imgName[2] = '(';
                              imgName[3] = '5';
                              imgName[4] = set[filterCha[k]/36];
                              if(k == numChaMatch){
                                    imgName[0] = target;
                                    //imgName[3] = '6'; // 60
                                    imgName[4] = '8';   // 58 //59 //60
                              }
                              imgName[5] = ')';
                              imgName[6] = '.';
                              imgName[7] = 'b';
                              imgName[8] = 'm';
                              imgName[9] = 'p';
                              imgResult = sprintf(inputName,
"%c%c%c%c%c%c%c%c%c%c", imgName[0], imgName[1], imgName[2], imgName[3],
imgName[4], imgName[5], imgName[6], imgName[7], imgName[8], imgName[9]);

                              img=cvLoadImage(inputName,
CV_LOAD_IMAGE_GRAYSCALE);
```

```
                        if(!img) {
                                printf("Could not load image file: %s\n",
inputName);
                                system("PAUSE");
                                return -1;
                        }
                        depth    = img->depth;
                        height     = img->height;
                        width      = img->width;
                        step       = img->widthStep;
                        channels   = img->nChannels;
                        data       = (uchar *)img->imageData;


                        //binarization
                        for(int i=0;i<height;i++)
                                for(int j=0;j<width;j++)
                                        if(data[i*step+j] <= 128)
                                                data[i*step+j] = 0;
                                        else
                                                data[i*step+j] = 255;

                        //find out boundary
                        for (int i=0; i < height; i++)
                                for (int j=0; j < width; j++)
                                        if(data[i*step+j] == 0)
                                                if(data[(i-1)*step+j] == 255)
                                                        data[(i-1)*step+j] = 100;
                                                if(data[(i+1)*step+j] == 255)
                                                        data[(i+1)*step+j] = 100;
                                                if(data[i*step+j-1] == 255)
                                                        data[i*step+j-1]  = 100;
                                                if(data[i*step+j+1]  == 255)
                                                        data[i*step+j+1]  = 100;

                        for (int i=0; i < height; i++)
                                for (int j=0; j < width; j++)
                                        if(data[i*step+j] == 0)
                                                data[i*step+j] = 255;

                        imgResult = sprintf(inputName, "%c%c%c%c%c%c%c",
character[filterCha[k]%36], '_', 'X', '.', 'b', 'm', 'p');
                        cvSaveImage(inputName,img);
```

```
                        //find startWidth and startHeight
                        int first2 = 0;
                        int line = 0;
                        while(first2 == 0){
                                int i = 0;
                                int j = line;
                                while((i <= line && j >= 0) && first2 == 0){
                                        if(data[i*step+j] == 100){
                                                startWidth = i;
                                                startHeight = j;
                                                first2++;
                                        }
                                        i++;
                                        j--;
                                }
                                line++;
                        }
                        //freeman chain code
                        img2=cvLoadImage(inputName,
CV_LOAD_IMAGE_GRAYSCALE);
                        if(!img2){
                                printf("Could not load image file: %s\n",
inputName);
                                system("PAUSE");
                                return -1;
                        }
                        data2 = (uchar *)img2->imageData;
                        int min;
                        int min2[50][3]={};
                        int min2Count = 0;
                        width = startWidth;
                        height = startHeight;
                        for(int i = 0; i < 300; i++)
                                for(int j = 0; j < 250; j++)
                                        library2[i][j] = 8;
                        while(start){
                                min = 256;
                                if((data[(width-1)*step+height+1] == 100 )
&& (data2[(width-1)*step+height+1] <= min)){
                                        if(min == 256)
                                                min = data2[(width-
1)*step+height+1];
                                        else if(sequence > 5){
                                                min2[min2Count][0] = width-1;
                                                min2[min2Count][1] = height+1;
                                                min2Count++;
                                        }
                                }
```

```
                                    if((data[(width-1)*step+height-1] == 100 ) &&
                                (data2[(width-1)*step+height-1]  <= min)){
                                            if(min == 256)
                                                    min = data2[(width-1)*step+height-
1] ;
                                            else if(sequence > 5){
                                                    min2[min2Count][0] = width-1;
                                                    min2[min2Count][1] = height-1;
                                                    min2Count++;
                                            }
                                    }

                                    if((data[(width+1)*step+height-1] == 100 ) &&
                                (data2[(width+1)*step+height-1] <= min)){
                                            if(min == 256)
                                                    min = data2[(width+1)*step+height-
1];
                                            else if(sequence > 5){
                                                    min2[min2Count][0] = width+1;
                                                    min2[min2Count][1] = height-1;
                                                    min2Count++;
                                            }
                                    }
                                    if((data[(width+1)*step+height+1] == 100 ) &&
(data2[(width+1)*step+height+1] <= min)){
                                            if(min == 256)
                                                    min =
data2[(width+1)*step+height+1];
                                            else if(sequence > 5){
                                                    min2[min2Count][0] = width+1;
                                                    min2[min2Count][1] = height+1;
                                                    min2Count++;
                                            }
                                    }
                                    if((data[width*step+height+1] == 100 ) &&
(data2[width*step+height+1] <= min)){
                                            if(min == 256)
                                                    min = data2[width*step+height+1];
                                            else if(sequence > 5){
                                                    min2[min2Count][0] = width;
                                                    min2[min2Count][1] = height+1;
                                                    min2Count++;
                                            }
                                    }
```

```
                                if((data[(width-1)*step+height] == 100 ) &&
                            (data2[(width-1)*step+height] <= min)){
                                        if(min == 256)
                                                min = data2[(width-
1)*step+height];
                                        else if(sequence > 5){
                                                min2[min2Count][0] = width-1;
                                                min2[min2Count][1] = height;
                                                min2Count++;
                                        }
                                }
                                if((data[(width)*step+height-1] == 100 ) &&
                            (data2[(width)*step+height-1] <= min)){
                                        if(min == 256)
                                                min = data2[(width)*step+height-
1];
                                        else if(sequence > 5){
                                                min2[min2Count][0] = width;
                                                min2[min2Count][1] = height-1;
                                                min2Count++;
                                        }
                                }
                                if((data[(width+1)*step+height] == 100 ) &&
(data2[(width+1)*step+height] <= min)){
                                        if(min == 256)
                                                min =
data2[(width+1)*step+height];
                                        else if(sequence > 5){
                                                min2[min2Count][0] = width+1;
                                                min2[min2Count][1] = height;
                                                min2Count++;
                                        }
                                }

                                if((data[(width-1)*step+height+1] == 100 ||
data[(width-1)*step+height+1] == 200) && (data2[(width-1)*step+height+1] ==
min) && !((sequence == 1 || sequence == 2) && startHeight == height+1 &&
startWidth == width-1)){
                                        for(int minC = 0; minC < 20; minC++)
```

```
                                if(min2[minC][0] == width-1 && min2[minC][1]
                        == height+1)
                                                min2[minC][2] = 1;
                                        data2[(width-1)*step+height+1]++;
                                        data[(width-1)*step+height+1] = 200;
                                        chainCode[sequence] = 1;
                                        library[1][k]++;
                                        width -= 1;
                                        height += 1;
                                }
                                else if((data[(width-1)*step+height-1] ==
                        100 || data[(width-1)*step+height-1] == 200) &&
                        (data2[(width-1)*step+height-1] == min)
                        && !((sequence == 1 || sequence == 2) &&
                        startHeight == height-1 && startWidth == width-
                        1)){
                                        for(int minC = 0; minC < 20; minC++)
                                                if(min2[minC][0] == width-1 &&
min2[minC][1] == height-1)
                                                        min2[minC][2] = 1;
                                        data2[(width-1)*step+height-1]++;
                                        data[(width-1)*step+height-1] = 200;
                                        chainCode[sequence] = 3;
                                        library[3][k]++;
                                        width -= 1;
                                        height -= 1;
                                }
                                else if((data[(width+1)*step+height-1] ==
                        100 || data[(width+1)*step+height-1] == 200) &&
                        (data2[(width+1)*step+height-1] == min)
                        && !((sequence == 1 || sequence == 2) &&
                        startHeight == height-1 && startWidth ==
                        width+1)){
                                        for(int minC = 0; minC < 20; minC++)
                                                if(min2[minC][0] == width+1 &&
min2[minC][1] == height-1)
                                                        min2[minC][2] = 1;
                                        data2[(width+1)*step+height-1]++;
                                        data[(width+1)*step+height-1] = 200;
                                        chainCode[sequence] = 5;
                                        library[5][k]++;
                                        width += 1;
                                        height -= 1;
                                }
```

```
                              else if((data[(width+1)*step+height+1] == 100
                        || data[(width+1)*step+height+1] == 200) &&
                        (data2[(width+1)*step+height+1] == min)
                        && !((sequence == 1 || sequence == 2) && startHeight
                        == height+1 && startWidth == width+1)){
                                    for(int minC = 0; minC < 20; minC++)
                                          if(min2[minC][0] == width+1 &&
min2[minC][1] == height+1)
                                                min2[minC][2] = 1;
                                    data2[(width+1)*step+height+1]++;
                                    data[(width+1)*step+height+1] = 200;
                                    chainCode[sequence] = 7;
                                    library[7][k]++;
                                    width += 1;
                                    height += 1;
                              }
                              else if((data[(width)*step+height+1] == 100 ||
data[(width)*step+height+1] == 200) && (data2[(width)*step+height+1] == min)
&& !((sequence == 1 || sequence == 2) && startHeight == height+1 && startWidth
== width)){
                                    for(int minC = 0; minC < 20; minC++)
                                          if(min2[minC][0] == width &&
min2[minC][1] == height+1)
                                                min2[minC][2] = 1;
                                    data2[(width)*step+height+1]++;
                                    data[(width)*step+height+1] = 200;
                                    chainCode[sequence] = 0;
                                    library[0][k]++;
                                    height += 1;
                              }
                              else if((data[(width-1)*step+height] == 100 ||
                        data[(width-1)*step+height] == 200) && (data2[(width-
                        1)*step+height] == min)&& !((sequence == 1 ||
                        sequence == 2) && startHeight == height && startWidth
                        == (width-1))){
                                    for(int minC = 0; minC < 20; minC++)
                                          if(min2[minC][0] == width-1 &&
min2[minC][1] == height)
                                                min2[minC][2] = 1;
                                    data2[(width-1)*step+height]++;
                                    data[(width-1)*step+height] = 200;
                                    chainCode[sequence] = 2;
                                    library[2][k]++;
                                    width -= 1;
                              }
```

```
                                else if((data[(width)*step+height-1] == 100
|| data[(width)*step+height-1] == 200) && (data2[(width)*step+height-1] ==
min) && !((sequence == 1 || sequence == 2) && startHeight == height-1 &&
startWidth == width)){
                                        for(int minC = 0; minC < 20; minC++)
                                                if(min2[minC][0] == width &&
min2[minC][1] == height-1)
                                                        min2[minC][2] = 1;
                                        data2[(width)*step+height-1]++;
                                        data[(width)*step+height-1] = 200;
                                        chainCode[sequence] = 4;
                                        library[4][k]++;
                                        height -= 1;
                                }
                                else if((data[(width+1)*step+height] == 100
|| data[(width+1)*step+height] == 200) && (data2[(width+1)*step+height] ==
min) && !((sequence == 1 || sequence == 2) && startHeight == height &&
startWidth == width+1)){
                                        for(int minC = 0; minC < 20; minC++)
                                                if(min2[minC][0] == width+1 &&
min2[minC][1] == height)
                                                        min2[minC][2] = 1;
                                        data2[(width+1)*step+height]++;
                                        data[(width+1)*step+height] = 200;
                                        chainCode[sequence] = 6;
                                        library[6][k]++;
                                        width += 1;
                                }
                                else{
                                        int minC = 0;
                                        int check = 0;
                                        while(minC < min2Count && check ==
0){
                                                if(min2[minC][2] != 1){
                                                        width = min2[minC][0];
                                                        height = min2[minC][1];
                                                        min2[minC][2] = 1;
                                                        check = 1;
                                                }
                                                minC++;
                                        }
                                }
```

```
                                    sequence++;
                                    if((width == startWidth && height ==
startHeight && sequence > 10 )|| sequence >= 1500)
                                          start = 0;
                              }
                              cvSaveImage(inputName,img);

                              //smoothing the chain code
                              int chain[8] = {0,1,2,3,4,5,6,7};
                              int chainCode2[500];
                              for(int i = 0; i < sequence; i++)
                                    chainCode2[i] = chainCode[i];

                              for(int i = 0; i < sequence-2; i++){
                                    int set[5];
                                    set[0] = chainCode2[i];
                                    set[1] = chainCode2[i+1];
                                    set[2] = chainCode2[i+2];
                                    set[3] = chainCode2[i+3];
                                    set[4] = chainCode2[i+4];
                                    if(set[0] == set[1]){
                                          if(set[3] == set[0] || set[4] == set[0])
                                                set[2] = set[0];
                                    }
                                    else if(set[3] == set[4]){
                                          if(set[0] == set[3] || set[1] == set[3])
                                                set[2] = set[3];
                                    }
```

```
                                else if(set[0] == set[4]){
                                        if(set[0] == 0)
                                        {
                                                if(set[1] == chain[set[0]+1] ||
set[1] == 0)
                                                        set[1] = set[0];set[2] =
set[0];
                                                if(set[3] == chain[set[0]+1] ||
set[3] == 0)
                                                        set[3] = set[0];set[2] =
set[0];
                                        }
                                        else{
                                                if(set[1] == chain[set[0]+1] ||
set[1] == chain[set[0]-1])
                                                        set[1] = set[0];set[2] =
set[0];
                                                if(set[3] == chain[set[0]+1] ||
set[3] == chain[set[0]-1])
                                                        set[3] = set[0];set[2] =
set[0];
                                        }
                                }

                                chainCode2[i] = set[0];
                                chainCode2[i+1] = set[1];
                                chainCode2[i+2] = set[2];
                                chainCode2[i+3] = set[3];
                                chainCode2[i+4] = set[4];
                        }
                        for(int i = 0; i < sequence; i++)
                                library2[i][k] = chainCode2[i];
                }
                //Edit Distance
                //get the chain code length of target
                int lenTarget = 0;
                int minED = 1000;
                int pos;
                for(int i = 0; i < 300; i++){
                        if(library2[i][numChaMatch] != 8)
                                lenTarget++;
                }
```

```
                    for(int p = 0; p < numChaMatch; p++){
                          float matrix[300][300];
                          int lenInput = 0;

                          //get chain code length of inputs
                          for(int i = 0; i < 300; i++){
                                if(library2[i][p] != 8)
                                      lenInput++;
                          }

                          int x, y, s1len, s2len;
                          s1len = lenTarget;
                          s2len = lenInput;
                          matrix[0][0] = 0.0;
                          for (x = 1; x <= s2len; x++)
                                matrix[x][0] = matrix[x-1][0] + 0.0;
                          for (y = 1; y <= s1len; y++)
                                matrix[0][y] = matrix[0][y-1] + 0.0;
                          for (x = 1; x <= s2len; x++){
                                for (y = 1; y <= s1len; y++){
                                      double num = 1.0;
                                      if(abs(library2[y-1][numChaMatch] -
library2[x-1][p]) == 0)
                                            num = 0;
                                      else if(abs(library2[y-
1][numChaMatch] - library2[x-1][p]) == 1 || abs(library2[y-1][numChaMatch] -
library2[x-1][p]) == 7)
                                            num = 0.2;
                                      else if(abs(library2[y-
1][numChaMatch] - library2[x-1][p]) == 2 || abs(library2[y-1][numChaMatch] -
library2[x-1][p]) == 6)
                                            num = 1.5;
                                      else if(abs(library2[y-
1][numChaMatch] - library2[x-1][p]) == 3 || abs(library2[y-1][numChaMatch] -
library2[x-1][p]) == 5)
                                            num = 2.0;
                                      else if(abs(library2[y-
1][numChaMatch] - library2[x-1][p]) == 4)
                                            num = 2.4;
```

```
                                        matrix[x][y] = MIN3(matrix[x-1][y] +
                                1, matrix[x][y-1] + 1, matrix[x-1][y-1] +
                                num);
                                }
                        }

                if(matrix[lenInput][lenTarget] < minED){
                                minED = matrix[lenInput][lenTarget];
                                pos = p;
                        }
                }

                if(character[filterCha[pos]%36] != '1' ||
character[filterCha[pos]%36] != 'B' || character[filterCha[pos]%36] != 'D'
|| character[filterCha[pos]%36] != 'K' || character[filterCha[pos]%36] !=
'M'){
                        printf("\nThe character is predicted to
be: %c\n",character[filterCha[pos]%36]);
                }
                else{
                        //Summation of Chain Code
                        //result comparing
                        int difference[9][250] = {0};

                        for(int i = 0; i < 8; i++)
                                for(int j = 0; j < numChaMatch; j++)
                                        difference[i][j] = abs(library[i][j]
- library[i][numChaMatch]);

                        //sum up all differences
                        for(int j = 0; j < numChaMatch; j++)
                                for(int i = 0; i < 8; i++)
                                        difference[8][j] = difference[8][j] +
difference[i][j];
                        //KNN, find minimum three differences
                        double min1 = 1000, min2 = 1000, min3 = 1000;
                        int one, two, three;
                        for(int i = 0; i < numChaMatch; i++){
                                if(difference[8][i] < min1){
                                        min1 = difference[8][i];
                                        one = i;
                                }
                        }
```

```
                        for(int i = 0; i < numChaMatch; i++){
                                if(difference[8][i] < min2 &&
(difference[8][i] >= min1 && i != one)){
                                        min2 = difference[8][i];
                                        two = i;
                                }
                        }
                        for(int i = 0; i < numChaMatch; i++){
                                if(difference[8][i] <= min3 &&
(difference[8][i] >= min2 && i != two && i != one)){
                                        min3 = difference[8][i];
                                        three = i;
                                }
                        }

                        double one2,two2,three2;
                        one2 = (0.1*min1) * (1.0-
ANN[c_sample][filterCha[one]%36]);
                        two2 = (0.1*min2) * (1.0-
ANN[c_sample][filterCha[two]%36]);
                        three2 = (0.1*min3) * (1.0-
ANN[c_sample][filterCha[three]%36]);

                        if((character[filterCha[one]%36] ==
character[filterCha[two]%36]) && (character[filterCha[one]%36] ==
character[filterCha[three]%36]))
                                printf("\nThe character is predicted to
be: %c\n",character[filterCha[one]%36]);
                        else if(one2 <= two2 && one2 <= three2)
                                printf("\nThe character is predicted to
be: %c\n",character[filterCha[one]%36]);
                        else if(two2 <= one2 && two2 <= three2)
                                printf("\nThe character is predicted to
be: %c\n",character[filterCha[two]%36]);
                        else if(three2 <= one2 && three2 <= two2)
                                printf("\nThe character is predicted to
be: %c\n",character[filterCha[three]%36]);
                    }
                }
        }
        system("pause");
        return 0;
}
```

fyp2_report

ORIGINALITY REPORT

11 %	9 %	7 %	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	bart.aero.psu.edu *Internet Source*	1%
2	www.zhciq.gov.cn *Internet Source*	1%
3	modely.jika.eu *Internet Source*	1%
4	en.wikibooks.org *Internet Source*	1%
5	www.utar.edu.my *Internet Source*	1%
6	www.opencv.org.cn *Internet Source*	< 1%
7	www.117766.cn *Internet Source*	< 1%
8	www.campsoftware.com *Internet Source*	< 1%
9	Ferreira da Conceicao, Thiago. "Corrosion protection of magnesium AZ31 allo... *Publication*	< 1%
10	www.sfitengg.org *Internet Source*	< 1%
11	Kautz, Stefanie(and Lumbsch, Thorsten). "Acacia-inhabiting Pseudomyrmex a... *Publication*	< 1%
12	www.mathworks.de *Internet Source*	< 1%
13	shop.illy.com *Internet Source*	< 1%
14	files.2-stars.net *Internet Source*	< 1%
15	blog.csdn.net	< 1%

	Internet Source	
16	robotix.in *Internet Source*	< 1%
17	www.cv.nrao.edu *Internet Source*	< 1%
18	www.sk.rs *Internet Source*	< 1%
19	Weisz, Gabriel, and James C. Hoe. "C-to-CoRAM : compiling perfect loop nes... *Publication*	< 1%
20	bbs.diyrobot.org.cn *Internet Source*	< 1%
21	Steeb, . "Bitwise Operations", Problems And Solutions In Scientific Computing... *Publication*	< 1%
22	Kazuo Tai. "The kinetics of hydrolytic polymerization of ε-caprolactam. II. Dete... *Publication*	< 1%
23	www.infinitecode.com *Internet Source*	< 1%
24	Y., Krishna.. "Acoustic Characteristics of Vowels in Telugu", Language in Indi... *Publication*	< 1%
25	International Journal of Pattern Recognition and Artificial Intelligence, 1989. *Publication*	< 1%
26	www.sachverstaendigenrat-wirtschaft.de *Internet Source*	< 1%
27	www.finance.hq.navy.mil *Internet Source*	< 1%
28	www.greystone.ca *Internet Source*	< 1%
29	M. T. Afzal. "Recognizing features from orthographic images using neural netw... *Publication*	< 1%
30	Behrawan, Houshang(Flügel, Wolfgang-Albert and Hochschild, Volker). "Hydr... *Publication*	< 1%
31	waterplanning.files.wordpress.com *Internet Source*	< 1%
32	teacher.buet.ac.bd *Internet Source*	< 1%
33	Moisidi, Margarita. "Geological geophysical and seismological investigations fo... *Publication*	< 1%

34	www.askrprojects.net *Internet Source*	< 1%
35	webapplicationsec.g.hatena.ne.jp *Internet Source*	< 1%
36	kubus.rulez.pl *Internet Source*	< 1%
37	Edwin Naroska. "A combined hardware and software architecture for secure c... *Publication*	< 1%
38	Ivits-Wasser, Eva. "Potential of remote sensing and GIS as landscape structu... *Publication*	< 1%
39	homes.esat.kuleuven.ac.be *Internet Source*	< 1%
40	Boldrini, Claudia <1978>(Viola, Erasmo). "Mixed Mode Fracture Behaviour of ... *Publication*	< 1%
41	Röth, Ernst-Peter. "Description of the anisotropic radiation transfer model ART... *Publication*	< 1%
42	www.soapandmore.com *Internet Source*	< 1%
43	qtc.jp *Internet Source*	< 1%
44	Stiefelhagen, Rainer. "Tracking and modeling focus of attention in meetings [on... *Publication*	< 1%
45	BALLARD, GREY; DEMMEL, JAMES; HOLTZ, OLGA and SCHWARTZ, ODE... *Publication*	< 1%
46	W-Y Wu. "Shape recognition using fuzzy string-matching technique", Imaging ... *Publication*	< 1%
47	Siala, Haytham(O'Keefe, B). "Cultural influences on consumer interactions in t... *Publication*	< 1%
48	Wynands, David (Prof. Dr. Christoph Brabec, Prof. Dr. Karl Leo and Technisc... *Publication*	< 1%
49	ro.uow.edu.au *Internet Source*	< 1%
50	SAIF ZAHIR. Journal of Circuits Systems and Computers, 2002 *Publication*	< 1%
51	Liu, Guojun(Fan, Z). "A study on twin-screw rheo-diecasting of AZ91D Mg-allo... *Publication*	< 1%

52	Kouba, Josef. "Investigation of silicon nitride based two-dimensional photonic … *Publication*	< 1%
53	Neumann, Volkmar. "Numerical modeling and phase prediction in deep overp… *Publication*	< 1%

EXCLUDE QUOTES	OFF	EXCLUDE MATCHES	OFF
EXCLUDE BIBLIOGRAPHY	OFF		