Handwritten Computer Program Recognition, Compilation & Execution Application

M.K Wanigapura

149234E

FACULTY OF INFORMATION TECHNOLOGY,

UNIVERSITY OF MORATUWA

2017

Handwritten Computer Program Recognition, Compilation & Execution Application

M.K Wanigapura

149234E

A DISSERTATION

SUBMITTED TO THE

FACULTY OF INFORMATION TECHNOLOGY,

UNIVERSITY OF MORATUWA, SRI LANKA

FOR THE PARTIAL FULFILLMENT OF THE REQUIREMENT

OF THE MASTERS DEGREE OF

MASTER OF SCIENCE IN INFORMATION TECHNOLOGY

JUNE 2017

Declaration

I hereby declare that the project work entitled "Handwritten Computer Program Recognition, Compilation & Execution Application", submitted to the university of Moratuwa, Sri Lanka, is a record of an original work done by me, under the guidance of my Supervisor Senior Lecturer Mr.Saminda Premaratne. This project work is submitted in the partial fulfillment of the requirement for the award of the degree of Master of Science in Information Technology. The results embodied in this report have not been submitted to any other University or Institution for the award of any degree or diploma. Information derived from the published or unpublished work of others has been acknowledged in the text and a list of references is given.

Supervisor

Mr.Saminda Premaratne…… Date: / /

(Senior Lecturer – University of Moratuwa)

Student

M.K Wanigapura…… Date: / /

Dedication

I would like to dedicate this thesis to my supervisor, senior lecturer **Mr.Saminda Premaratne** who gave me tremendous support and motivation throughout the entire process of the research.

Acknowledgement

I would like to express my special appreciation and thanks to my advisor, Senior Lecturer Mr.Saminda Premaratne, you have been a tremendous mentor for me. I would like to thank you for encouraging my research and for allowing me to grow as a research student. Your advice on both research as well as on my career has been invaluable.

Besides my supervisor, I would like to thank, Prof. Asoka S. Karunananda Dean of Kothalawala Defense University. The quality of this thesis and research have improved by using comprehensive guidelines of Prof. Karunananda. I would also like to acknowledge the rest of lectures in the department of IT at university of Moratuwa, Sri Lanka for their encouragement.

Last but not the least, I would like to thank my colleagues and family for supporting me spiritually throughout and my life in general.

Abstract

Make a computer program that does automatically recognize a handwritten computer program, compile and execution is extremely difficult. The reasons for these are the various types, shapes of handwritten characters of different peoples. This paper provides an accurate handwritten character recognition method, which blend with the image processing and the training for handwritten characters. Further, automatic error correction of handwritten program, compilation and execution of the program, will be discussed in detail throughout the paper.

Recognition of handwritten characters has been a research challenged. Handwritten character recognition belongs to the family of optical character recognition performing automatic identification. Among other issues in optical character recognition, handwritten identification, accuracy or the correctness is the main research issues.

This research has developed a system for identification of handwritten characters. The solution can identify the handwritten characters with a higher level of accuracy. This solution can deal with any images captured by a digital camera or scanned images of handwritten characters. The algorithm applies on preprocessed image with handwritten characters. The preprocessor has used standard image processing techniques and the trainer is used to train for the specific handwritten characters.

Table of Contents

Table of Figures

Table of Tables

Chapter 1

1 Introduction

1.1 Prolegomena

Optical Character Recognition (OCR) is a conversion of scanned or printed text images, handwritten text images into machine encoded text for further processing. It is generally an offline process which can be done by electronically. In the past, the OCR has its limits in just reading the printed fonts. However, as newer advanced stages, handwriting character recognition OCR is developed.

Handwriting recognition is the ability of a computer to receive and interpret handwritten input from sources such as paper documents, photographs, touch-screens and scanned images, etc.... Handwritten recognition has been studied for nearly forty years and there are many proposed approaches. With the use of handwritten character recognition, it will enable to provide the clearest data to operational workflow. In the handwriting recognition process of an image containing text must be appropriately fed and preprocessed. After image acquisition and preprocessing, the text must undergo the process of language training. Small processed pieces of the text will be the result and these pieces put into a training and recognition process by the system.

Research gap or the research problem in this study is to find out the areas of inadequate attention given to the handwritten computer program recognition and introduce a new system for identifying the handwritten computer program. Our hypothesis is addressing the problem can be solved by introducing more accurate pipeline of preprocessing and use the ‘Tesseract’ open source OCR library in novel way.

Newly introduced system ensures the multiple input forms of images (JPEG, PNG), high level of accuracy of character identification and detects programmatically errors automatically. The system is applicable in scenarios such as assignment marking, paper marking, etc...

1.2 Background and motivation

Handwriting recognition of characters has been around since the 1980s.The task of handwritten digit recognition, using a classifier, has great importance and use such as – online handwriting recognition on computer tablets, recognize zip codes on mail for postal mail sorting, processing bank check amounts, numeric entries in forms filled up by hand (for example tax forms) and so on. There are different challenges faced while attempting to solve this problem. The handwritten characters are not always of the same size, thickness, or orientation and position relative to the margins.

Written examinations and assignments are the most common student evaluation methodologies in the present education system. This makes heavy work for the invigilators because they should mark the answer scripts of each and every student.

When it comes to the computing subjects, lecturers and teachers have to put more effort on marking answer scripts with program codes. They have to go through the line by line of the code in order to identify the errors and final output.

Thus, this issue can be overcome by automating the whole process of marking by hand written computer codes. The goal was to implement a system to automatically recognize the handwritten Java program, compile and run the program.

1.3 Problem statement

Above section described about the various research areas in OCR as well as handwritten character recognition, and background information on existing OCR methodologies using different technologies. Chapter 2 has included a comprehensive literature review of optical character recognition (OCR). In depth study of OCR reveals that there are many aspects of handwritten character recognition, still not resolved properly. Especially currently available OCR engines are not capable of identification of various types of handwritten characters of different people. The main problems with those characters are different size, shape and the different writing style.

Therefore, the research problem can be defined as inaccurate or the inefficient method of handwritten computer program identification and less attention for the better prepossessing of images before feeding to the recognition process.

1.4 Hypothesis

It is hypothesized that the above addressed problem can be solved by introducing novel and more efficient method for handwritten computer program identification using Image processing and Tesseract open source OCR engine.

1.5 Objectives

(i) Critically review the state of the art of optical character recognition and Tesseract open source OCR engine

(ii) To do in depth study of optical character recognition algorithms with a particular emphasis on image processing techniques – effective preprocessing

(iii) To develop a new system for handwritten computer program recognition

(iv) To evaluate the performance of the new system

1.6 Blend of Image Processing & Training approach

According to the proposed new approach entire process has divided into three major phases which are the, preprocessing of the image, program code identification (handwritten identification), post processing with the error correction and execution of the identified program.

The main purpose of identification of the handwritten computer program using the OCR method is to identify the program syntaxes accurately from the uploaded image file. While the second phase expected to compile and execute the identified Java code successfully in order to have expected outcome of the program. Identification of handwritten Java code (program) is a challenging task, and it should be more accurate

than the other previous methods. Preprocessing is an important part of the recognition process and it must be used correct image processing techniques in order to have well preprocessed image which should undergo the training process.

1.7 Structure of the thesis

The rest of the thesis is organized as follows. Chapter 2 critically reviews the literature on optical character recognition and handwritten character recognition as well as the Tesseract open source OCR engine and identifies the research problem. Chapter 3 is about the image processing techniques and the Tesseract open source OCR engine for recognition of handwritten characters in an image. Chapter 4 represents a new approach to identify the handwritten Java program. Chapter 5 and chapter 6 describe the design and implementation respectively. Chapter 7 is an evaluation of the new system. Chapter 8 concludes the research with a note on future work

1.8 Summary

This chapter gave an overall picture of the entire project presented in this thesis. It is included the background/motivation, problem definition, hypothesis, objectives and a brief overview of the solution. Next chapter presents a critical review of literatures on optical character recognition and the handwritten character recognition using image processing and the Tesseract open source OCR engine.

Chapter 2

2 Developments and Challenges in Handwritten Character Identification

2.1 Introduction

Chapter 1 gave a comprehensive description of the overall project described in this thesis. This chapter provides a critical review of the literature in relation to developments and challenges in Optical Character Recognition. For this purpose the review of the past researches has been presented under three major sections. Namely, early developments, modern trends and future challenges. At the end, this chapter defines the research problem as the inadequate use of combination of two or more methodologies (techniques) in order to accomplish higher rate of accurate recognition and lack of research works on identification of computer program syntax. In order for achieving higher accuracy rate blend of image processing and training methodology has been identified.

2.2 Categorization of OCR

OCR systems can be grouped into two categories, namely task-specific readers and general purpose page reader [1][2]. A task-specific reader handles only specific document types. Some of the most common task-specific readers process standard forms, bank checks, credit card slips, etc. These readers usually utilize custom made image lift hardware that captures only a few predefined document regions. For example, a bank check reader may just scan the courtesy amount field and a postal OCR system may just scan the address block on a mail piece. Such systems emphasize high throughput rates and low error rates. Rasmussen and his colleagues have done study of pipeline for handwritten form fields from an electronic health record [3]. They have presented an optical character recognition processing pipeline, which leverages the capabilities of existing third-party optical character recognition engines, and provides the flexibility offered by a modular custom developed system.

The system was configured and run on a selected set of form fields extracted from a corpus of handwritten ophthalmology forms. The specific task was to identify cataract type and severity for subjects using purely automated methods, which would be used for a genome-wide association study. Because the goal of the driving study was to find a limited number of 'clean' cases from among a large population, the primary performance goal was a high positive predictive value at the expense of sensitivity. Lexical post-processing optimization for handwritten word recognition is famous OCR technique [4]. The aim of this [4] work is to explore the combination of different lexical post- processing approaches in order to optimize the recognition rate, recognition time and memory requirements. Mathematical formulas extraction [5], [6] is one of major application of OCR. Mathematics has been widely applied in many fields, and it is more accurate than any other languages in describing information. Therefore, numerous mathematical formulas exist in all kinds of documents. There is no doubt that automatic mathematical formulas processing is very important and necessary, of which extract formulas from document images is the first step. In this paper [5], formulas extraction methods which are not based on recognition results are presented: isolated formulas are extracted based on Parzen-window [7] and embedded expressions are extracted based on 2-D structures detection. Parzen-window density estimation is essentially a data interpolation technique.

Many commercial general purpose OCR systems started as university projects, shareware or free-ware programs, and developed into sophisticated high quality OCR engines, software development kits, and ready to use software products meeting the high expectations of todays OCR market [1]. In [8] offer a perspective on the performance of current OCR systems by illustrating and explaining actual OCR errors made by three commercial devices. After discussing briefly the character recognition abilities of humans and computers, they have presented illustrated examples of recognition errors.

2.3 Essential Image Processing Techniques used in OCR

Machine simulation of human functions has been a challenging research field since the advent of digital computers. In OCR areas, which require a certain amount of

intelligence, On the other hand, humans still out-perform even the most powerful computers in the relatively functions such as vision. The study investigates the direction of the Optical Character Recognition research, analyzing the limitations of methodologies for the systems which can be classified based upon two major criteria. The data acquisition process (on-line or off-line) and the text type (machine- printed or hand-written). In generally there are five major stages in the OCR systems [9]: 1.Preprocessing, 2.Segmentation. 3. Feature Extraction, 4.Recognition, 5.Post processing.

Preprocessing is a common name for operations with images at the lowest level of abstraction both input and output are intensity images. The aim of preprocessing is an improvement of the image data that suppresses unwanted distortions or enhances some image features important for further processing. **Preprocessing** functions involve those operations that are normally required prior to the main data analysis and extraction of information, and are generally grouped **as radiometric or geometric corrections**. Some **standard correction** procedures may be carried out on the ground station before the data is delivered to the user. These procedures include radiometric correction to correct for uneven sensor response over the whole image and geometric correction to correct for geometric distortion due to Earth's rotation and other imaging conditions. Image preprocessing technique is used to increase the excellence of image quality for easy and efficient processing in next steps. Handwriting analysis needs to perform preprocessing steps such as binarization [10] and noise removal etc. In this [11] proposed method Salt and Pepper noise is removed using a median filter technique and Otsu thresholding technique is used for image binarization.

It is one the most important process that decides the success of character recognition technique. It is used to decompose an image of a sequence of characters into sub images of individual symbols by segmenting lines and words. After measuring the witting pressure, text lines are segmented from the binary document image. The present work implements a modified horizontal projection method of image that can segment individual text line from the previous and following text lines based on a rising section of the horizontal projection histogram of a document image. English handwritten document image, most of the time no gaps are present between two lines, which may create incorrect line segmentation due to overlapping between two lines if

a simple horizontal projection histogram is concern. In [11] proposed method, after creating the horizontal projection histogram of a binary document image, count the number of rising section and height of each rising section. The average height of the rising sections is treated as the threshold. Then consider each and every rising section and check the height of that rising section is greater than or equals to the threshold or not. If yes then based on that rising section of the horizontal histogram, the line is individually segmented from the actual binary document image, otherwise neglect that rising section as a false line segment.

The feature extraction is a most important part of any recognition system. HOG transformation [9],[12] is popular method to feature extraction. Transforming the input data into the set of features is called as feature extraction. When performing analysis of handwritten Marathi characters data one of the major problems is the number of characters involved. The minimum number of features handwritten characters that are useful in identifying in pattern classes. HOG transformation has used to detect and extract feature of handwritten characters.

Image recognition is the process of identifying and detecting an object or a feature in a digital image or video. This concept is used in many applications like systems for factory automation, character recognition, and security surveillance. Typical image recognition algorithms include: Optical character recognition, Pattern and gradient matching, Face recognition, and Scene change detection. The output image from segmentation phase goes to character recognition phase that consist of several steps. Michael Ryan et al. Have conducted a research about OCR, by using template matching [13], [14], [15]. Template matching is a technique in digital image processing for finding small parts of an image which match a template image. It can be used in manufacturing as a part of quality control, a way to navigate a robot, or to detect edges in images. Letter recognition is the foundation of the human reading system. Despite this, it tends to receive little attention in computational modelling of single word reading. Here [12] presented a model that can be trained to recognize letters in various spatial transformations. Degraded letters confusion errors that correlate with human controllability data. Analyses of the internal representations of the model suggested that a small set of learning visual feature detectors support the recognition of both upper case and lower case letters in various fonts and

transformations.

Chinese character recognition is an important branch of pattern recognition [16], it has important theoretical significance and using values in the most of technology fields. Based on the nearest neighbor algorithm [17], combined with pattern recognition, for offline handwritten Chinese character features, double weights elliptical neuron network as the basic unit, the combination of several basic units to offline handwritten Chinese character images of the most good coverage, therefore you can essentially eliminate the distortion of Handwritten Chinese influence on the stroke extraction. In [18] describe mathematical notation as a hierarchical structure of nested baselines. A baseline is a list which represents a horizontal arrangement of symbols in the expression. Each symbol has links to other baselines, which satisfy the spatial relations.

While high-accuracy character recognition has been achieved, in some applications even the few errors which are made are extremely costly. This study [19] perform both recognition and confidence measurement at the character level, so hypothesis graphs here are simply lists of single-character hypotheses, precluding the need for certain complex confidence measures such as multi-level word confidence derived from character- and word-level parameters or lattice-based parameters.

Normally a scanner scans a page as an image. Here [20] the problem is to identify the signature from a scanned image. It is difficult because there can be many other text and patterns in that scanned image. Hence it is required to set a signature area, which will help to identify the exact boundary of the signature in that scanned image. This research has proposed the novel method to signature recognition system.

The noise, introduced [21] by the optical scanning device or the writing instrument, causes disconnected line segments bumps and gaps in lines, filled loops etc. The distortion including local variations, rounding of corners, dilation and erosion, is also a problem. Prior to the character recognition, it is necessary to eliminate these imperfections. At the [21], the use of a shock filter in image processing mainly applies to the enhancement and restoration of degraded images. A shock filter is applied to an image. Finding a means of characterizing the shock locations will enable us to provide

thinner characters. For example, the selected criterion in enhancement processing is the curve concavity. The shocks are localized wherever the curve concavity changes. As such, the blur of the image's contours is removed while the curve is moving with a speed dependent on the local characteristics of the curve. The result is a shock created to separate the convex and concave parts of the intensity curve.

Yefeng Zheng et al. [22] have addressed the problem of the identification of text from noisy documents. They have segmented and identified handwriting from machine printed text and handwriting in a document often indicates corrections, additions or other supplemental information that should be treated differently from the main or body content, and the segmentation and recognition techniques for machine printed text and handwriting are significantly different. This approach has treat noise as a separate class and model noise based on selected features.

In recognition of machine printed characters on printed documents, a high accuracy can be obtained due to a priori regularity within a limited range of fonts known to an OCR system. Here [23] introduces a multifont classification scheme to help recognition of multifont and multisize characters. It uses typographical attributes such as ascenders, descenders and serifs obtained from a word image.

Handwritten document may originally be skewed or skewness may introduce in document scanning process. This effect is unintentional in many real cases, and it should be eliminated because it dramatically reduces the accuracy of the subsequent processes, such as segmentation and classification. Skewed lines [24] are made horizontal by calculating skew angle and making proper correction in the raw image.

The boundary detection of image is done to enable easier subsequent detection of pertinent features and objects of interest. Thinnin [25], [26] is a common technique using boundary detection of image. Purpose of Convert binary shapes obtained from edge/boundary detection or thresholding to 1-pixel wide lines. For example, the thresholded version of handwritten or printed alphanumerics can be thinned for better representation and further processing.

Wenxiao Du at Stanford University [27] has introduces "Code Runner", an Android

application that can recognize and execute handwritten code by users. Current prevalent OCR-engines can recognize printed text with high accuracy, but can hardly handle hanwritten text very well. The difficulty of handwritten text recognition is due to variation of characters and poor alignment of text line. Therefore, to achieve a workable solution for hand-written code recognition system, First customizes and train one popular OCR-engine, Tesseract, to make it able to deal with my own handwriting. Besides, several image processing methods and text-level post-processing algorithms have been adopted to enhance the system accuracy

2.4 Supporting Tools and Techniques in OCR

Handwritten signature is the most widely accepted biometric to identity verification. The target of research is to present online handwritten signature verification system based on discrete wavelet transform (DWT) [28] features extraction and feed forward back propagation error neural network recognition. Steps for verifying online handwritten signature in this system start with extracting pen position data (x and y positions) of points that forming the signature. Pen-movement angles are then derived from pen position data. To reduce variations in pen-position and pen-movement angles dimensionality, data are normalized and re-sampled. To enhance the difference between a genuine signature and its forgery, the signature is verified in DWT domain.

An effective Optical Character Recognition (OCR) system would be helpful to solve these issues. But the next question may arise that whether a single OCR system will be sufficient for encoding both handwritten and printed text or not. As printed characters generally, have uniform shape and structure, encoding them is less challenging in comparison with their handwritten counterpart. This is because of the fact that the shapes and structures of the handwritten characters vary from writer to writer. Even in a document written by a single writer these variations are sometimes are very distinct. So, the feature selection and classifier design are quite different for printed and handwritten character recognition. Even in the character segmentation and recognition steps, the characters of the said categories show a different dimension of complexities. Otsu’s [29] method , depicts a graphical representation for estimated threshold values by Otsu's method for different handwritten and printed word images. The graph shows that this feature also has significant contribution to the classification

of handwritten and printed word images.

A similarity measure neural network is used to identify characters and similarity measure compares the features of characters and the features of the indicators associated with the characters. In [30] has proposed a method for recognizing English characters in different fonts. The proposed method based on neural network is resistant to font variant. When the samples in new fonts are added to the database, the accuracy of existing methods rapidly decreases and they are not resistant to font variant, but to the accuracy of the proposed method that almost stays constant and does not much decrease. They have used similarity measure instead of distance measure in the SOM neural network because a person learns font-independent and a literate can read without knowing the font of the written note.

Tesseract [31] is an open-source OCR engine that was developed at HP between 1984 and 1994. Like a supernova, it appeared from nowhere for the 1995 UNLV Annual Test of OCR Accuracy, shone brightly with its results, and then vanished back under the same cloak of secrecy under which it had been developed. Now for the first time, details of the architecture and algorithms can be revealed. Labs, Bristol, and gained momentum as a possible software and/or hardware add-on for HP's line of flatbed scanners. Motivation was provided by the fact that the commercial OCR engines of the day were in their infancy, and failed miserably on anything but the best quality print.

2.5 Problem Definition

The literature review has identified various unsolved problems, including efficiency, and accuracy of character recognition as well as inadequate attention on the identification of handwritten computer programming language syntaxes.

It is evident from the literature that optical character recognition has been done or used in several domains, applications and systems.

As revealed by the literature, accuracy of classification algorithms and a combination of two or more techniques has been a hot topic and many researchers have identified the need for further research in this regard.

According to the history, despite many solutions are available for optical character recognition, they are rather needed more computations and training using many ways. Therefore, It needs to address the above problem by blending two or more technologies in order to achieve higher accuracy and develop a system, which needs less computational overhead. This is because; many researchers have shown suitability of a combination of two or more technologies.

2.6 Summary

This chapter presented a comprehensive literature review on the optical character recognition and identified the research problem as the inadequate use of combination of two or more methodologies (techniques) and lack of research works on identification of computer program syntax. Blend of image processing and training techniques has been identified to address the above problem. Next chapter will discuss the technology to be used for the proposed solution.

Chapter 3

3 Image processing and Tesseract OCR engine in handwritten computer program recognition

3.1 Introduction

A comprehensive review of the historical development of optical character recognition using different methods and technologies was given in the previous chapter. After the comprehensive study, it is decided to use the image processing with the Tesseract OCR for the identification of handwritten computer program.

3.2 Image processing – Preprocessing

Image Processing is a technique to enhance raw images received from various sources such as cameras, sensors, scanners, etc. Various techniques have been developed in Image Processing during the last four to five decades. Most of the techniques are developed for enhancing images obtained from various sources. Image preprocessing is one of the so called image processing technique.

As the image preprocessing techniques, this project has used gray scaling, thresholding, skewing, noise removing and thining in order to enhance the quality of handwritten image.

3.3 Tesseract OCR engine

Tesseract is an open-source OCR engine that was developed at HP between 1984 and 1994.) Tesseract was probably the first OCR engine able to handle white-on-black text so trivially. Tesseract never needed its own page layout analysis. Tesseract therefore assumes that its input is a binary image with optional polygonal text regions defined.

Tesseract recognition proceeds as a two-pass process. In the first pass, an attempt is made to recognize each word in turn. Each word that is satisfactory is passed to an adaptive classifier as training data. The adaptive classifier then gets a chance to more accurately recognize text lower down the page.

Since the adaptive classifier may have learned something useful too late to make a contribution near the top of the page, a second pass is run over the page, in which words that were not recognized well enough are recognized again. A final phase resolves fuzzy spaces, and checks alternative hypotheses for the x-height to locate smallcap text.

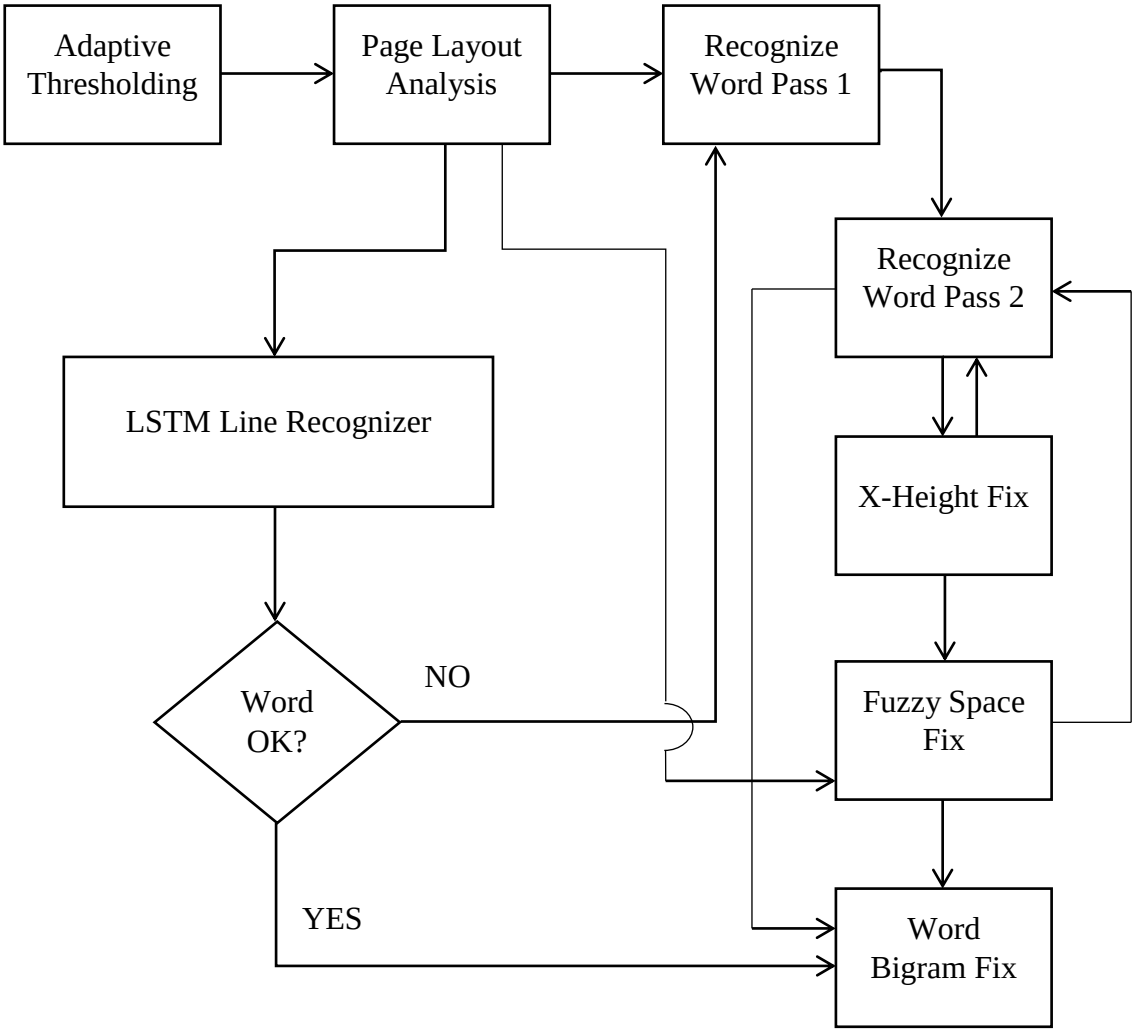


Figure 1 - Tesseract OCR engine

3.4 Summary

This chapter presented the simple overview of the main technologies used to develop the novel solution, which are image processing and the simple overview of the Tesseract OCR engine. Next chapter will include the approach which is going to be used the technologies describe here.

Chapter 4

4 Approach

4.1 Introduction

Having defined the problem in chapter 2, presented technology required for the proposed solution in chapter 3. The approach is described under the hypothesis, input to the system, the output of the system, process to convert the input to the output overall features of the system and users.

4.2 Hypothesis

It is hypothesized that the highly efficient handwritten computer program recognition can be achieved by introducing a blend of image processing and ‘Tesseract’ open source OCR engine training method.

4.3 Users

There are a number of users who can be benefited by the new application. More importantly, college lecturers, teachers, invigilators, examiners can be benefited from this solution. Those who are interested in study of handwritten computer program identification can also use this system for learning purposes.

4.4 Input

The main input for the system is the image of a handwritten computer program and the image of the training set. Image for the training set has included all the lowercase and uppercase characters in the English alphabet, numbers, and symbols. The system can accept input from various devices including scanners, smart phone cameras, and

digital cameras and the input could be a scanned image or any image file. The images submitted to the system should be in the original form of the image.

Scanner or digital camera much needed for taking the image of the handwritten program. Image processing methodologies for preprocessing of the image and the Tesseract open source OCR libraries will be used as adopted techniques. The Java compiler will be an another main input.

4.5 Output

The output of the system would be available in the computer as a successfully compiled and executed computer program or the text file which shows the programmatically errors. The output could be the perceivable output which is expected from the computer program such as printed message, alert message, the result of a calculation, view an image, play an audio file, create a file, etc. (graphical, textual, voice, images, videos, etc.)

4.6 Process

The entire process has divided into five major parts which are the preprocessing of the image, language training, character recognition, post processing and program execution for getting the expected result. The main purpose of image preprocessing is to make a row image as most readable one. Tesseract mostly used for reading printed text, therefore we needed to train the Tesseract, in order to getting identified the handwritten text. For that language training has been done as the second most important module of the project. After the training process, character recognition has been done in identifying the characters of the acquired image. Post processing needed to be done in order to correct the miss identified letters or the syntaxes and correct the programmatically errors and shows the programmatically errors as well. A final text file saved as a '.java' file and then pass it to the Java compiler for compilation. Then execute the compiled code for get the expected result of the Java program

The schematic diagram of the system is shown in Fig.2

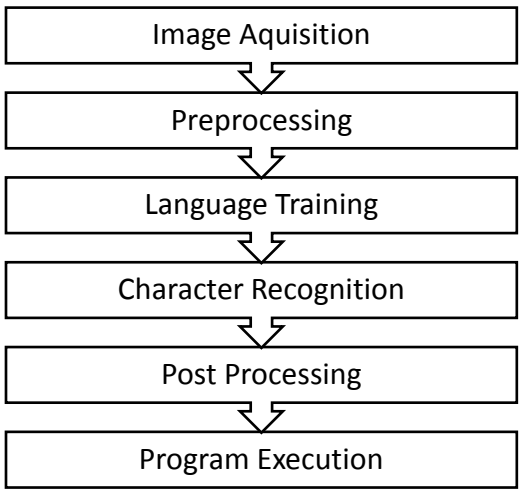


Figure 2 - schematic diagram of the recognition system

4.6.1 Image Acquisition

Most Important initial phases in OCR is to gather the image. In Image acquisition, the recognition system acquires scanned image as an input image. This image is acquired through a scanner, digital camera or any other suitable digital input device. Data samples for the experiment have been collected from different individuals. In the system user facilitates through the user interface for upload an image file which is consisting of hand written Java program.

4.6.2 Image Preprocessing

The pre-processing is a series of operations performed on the scanned input image. It essentially enhances the image rendering and it is suitable for language training. The various tasks performed on the image on pre-processing stage are shown in Fig.3. The Gray scaling process converts an image into a gray scaled image. Binarization process converts a grayscale image into a binary image using Otsu's global thresholding technique. Remove noises in the binarized image using median filtering, Zhang-Suen'' thinning algorithm used to thin a black and white image and finally skewing algorithm is applied for detecting and correcting skew and produces the pre-processed image suitable for language training.

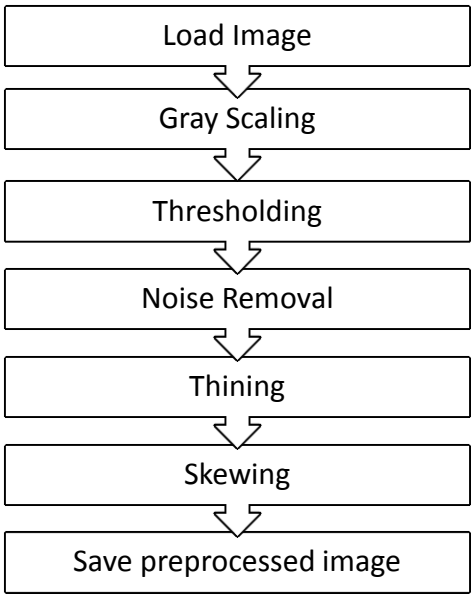


Figure 3 - Image Pre-processing Steps

4.6.2.1 Gray Scaling

First convert raw image to bitmap image and using the 'LockBits' method, it converted into the bitmap data. Use the LockBits method to lock an existing bitmap in system memory, therefore that it can be changed programmatically.

4.6.2.2 Otsu thresholding

Otsu's thresholding method used iterating through all the possible threshold values and calculating a measure of spread for the pixel levels, each side of the threshold, i.e. the pixels that either fall in the foreground or background. The aim was to find the threshold value where the sum of foreground and background spreads is at its minimum.

4.6.2.3 Noise Removal

Median filtering is a nonlinear method used to remove noise from images. It is widely used as it is very effective at removing noise while preserving edges. The median

filter works by moving through the image pixel by pixel, replacing each value with the median value of neighbouring pixels. The pattern of neighbours is called the "window", which slides, pixel by pixel over the entire image 3 pixel, 4 pixel and 5 pixel over the entire image. The median was calculated by first sorting all the pixel values from the window into numerical order, and then replacing the pixel being considered with the middle (median) pixel value.

4.6.2.4 Thining

Here two sub-iterations. In the fist one, a pixel s(x , y) is deleted if the following conditions are satisfied:

1. Its connectivity number is one.
2. It has at least two black neighbors and not more than six.
3. At least one of s(x,y+1), s(x-1,y) and s(x,y-1) are white.
4. At least one of s(x-1,y), s(x+1,y), and s(x,y-1) are white.

In the second sub-iteration the conditions in steps 3 and 4 change.

1. Its connectivity number is one.
2. It has at least two black neighbors and not more than six.
3. At least one of s(x-1,y), s(x,y+1), and s(x+1,y) are white.
4. At least one of s(x,y+1), s(x+1,y), and s(x,y-1) are white.

At the end, pixels satisfying these conditions will be deleted. If at the end of either sub-iteration there are no pixels to be deleted, then the algorithm stops.

4.6.2.5 Skewing

For deskew the skewed image used the following four steps:

- Find reference lines in the image.
- Calculate the angle of the lines.
- Calculate the skew angle as an average of the angles.
- Rotate the image.

The lines are detected with the Hough algorithm. Each point in the image can lie on an infinite number of lines. To find the reference lines, let each point vote for all the lines that pass through the point. The lines with the highest number of points are used as reference lines.

4.6.3 Language Training

There are two main steps in language training, Create test language using 'boxing' and Train for the new test language

In order to create the test language, image should be saved in '.tif' format. Using that .tif image box file generated and input that into the working directory manually.

Training language should be denoted by three letter acronym. Therefore the box file's and the .tif image file's naming convention should be in the correct format. This naming convention has described in detail in the implementation chapter.

Referred box file created in the working directory checked for the correct boxing. For the box file checking and the correction, one of the available box editors was used. After the creation of box file, font properties generated for the new language.

As training is the second main step 'unicharset' file and the 'master shape table' created and passed those two files into the tesseract 'mtraining' and the 'cntrainig, processes. In the training process 'inttemp', 'normproto', 'pffmtble' and the 'shapetable' need to create with respect to the language name.

Finally, these four files combined in order to complete the language training. The new language file can be found in the working directory.

4.6.4 Character Recognition

The created training data set has fed to the character recognition application as the training set of the project. Training set name or the language file name has given as the variable in the character recognition program.

```
var ocrEngine = new TesseractEngine(@"./tessdata", @"ntl", EngineMode.Default);
var imageWithText = Pix.LoadFromFile(imagePath);
var extractedText = ocrEngine.Process(imageWithText);
```

Figure 4 - Image feed for OCR

Test images which are needed to be read through the character recognition program can be given as the input to the system as shown in Fig.4 and, that images must be going through the preprocessing steps described above. Finally, the preprocessed image put into the Tesseract OCR engine.

4.6.5 Post Processing and program execution

The recognized handwritten computer program has taken as the output from the Tesseract OCR engine and so called output has put into the post processing. In the post processing, image has gone through the five consecutive steps as mentioned below.

- Format the code
- Check brackets
- Check Java syntaxes
- Correct the error syntaxes
- Save and execute the program

4.7 Features

In connection with the input, output, users and process, the overall features of the system include the following characteristics.

- Multiple input forms of images (JPEG, PNG,TIF, etc..)
- High level of accuracy
- User friendly
- Detect programmatically errors automatically
- Reliability
- High performance
- Research platform for researchers
- Improve the quality of handwritten codes

4.8 Summary

The chapter describes overall solution. Problem definition and assumption of the solution to that problem have mentioned. Clearly described inputs to the system, the outputs of the system and how convert input to output and features of novel solution. Next chapter will describe in detail, extended design of our process and what the system does.

Chapter 5

5 Design

5.1 Introduction

The previous chapter gave a full picture of the entire solution. This chapter describes the design of the solution for the process presented on the approach. The solution has been developed as a standalone application. This chapter describes the top-level architecture of the design by elaborating the role of each component of the architecture.

5.2 Top Level Architecture

The top level architecture of the application comprises of several modules, namely user interface module, preprocessing module, language training module, OCR module, and post processing module. Figure 5.1 illustrates the top level architecture.

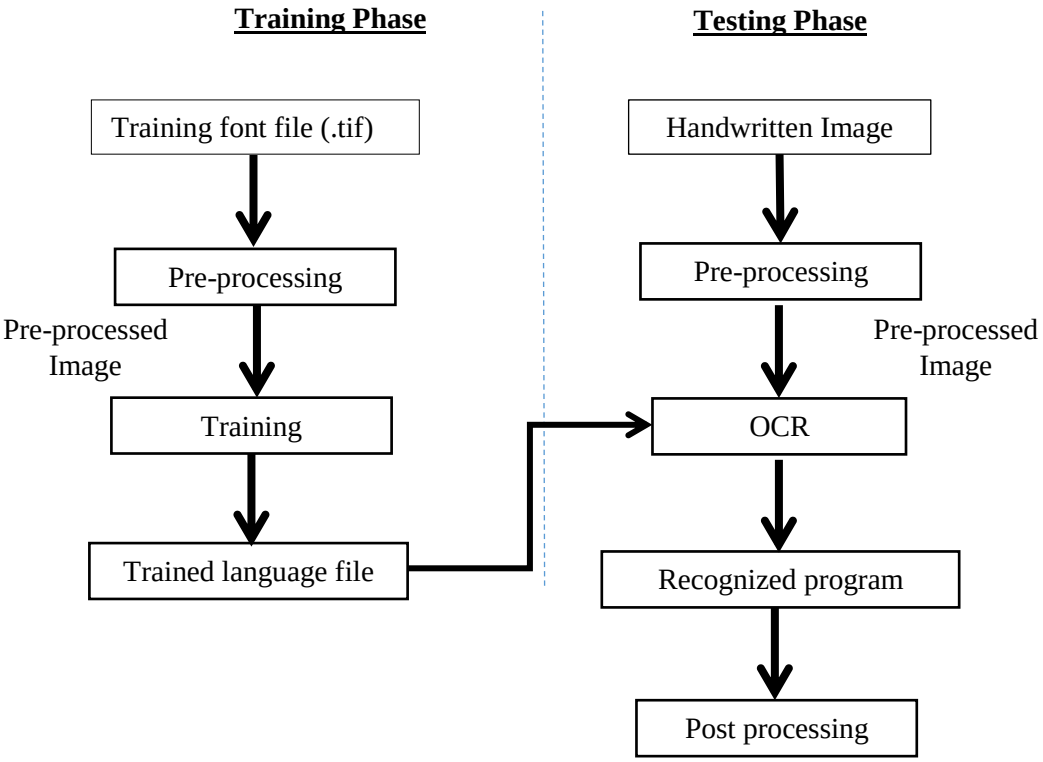


Figure 5 - Top Level Architecture

5.3 Interface Module

This module gives access to the system. It enables interacting with the system through a desktop client. Through this module image uploading and separate preprocessing steps are provided. The interface module offers facilities for entering input images and also receiving output images which are relevant to the particular preprocessing step. For an example: if input an original image first step of preprocessing – gray scaling, gives a gray scaled image as an output image.

5.4 Preprocessing module

The aquired handwritten image is going through the following preprocessing techniques. In design each and every step facilitates through a separate button. The design help users to change thresholding level according to the situation and select the optional values for the filtering as well.

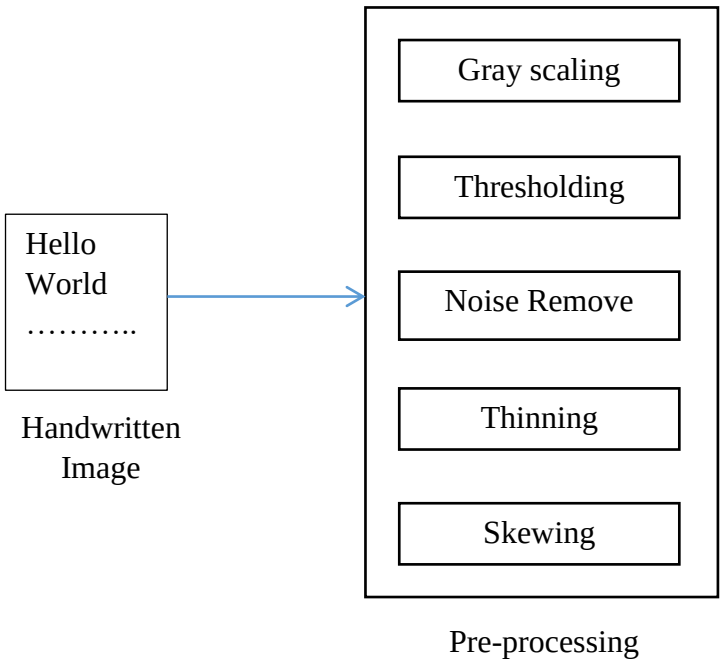


Figure 6 - Design of Pre-processing module

5.5 Language Training module

The language training interface has the functionalities of creating box file, create font properties and the create unicharset file. For correction of the box file it needs a separate box editor interface which can taken from the third party box editor.

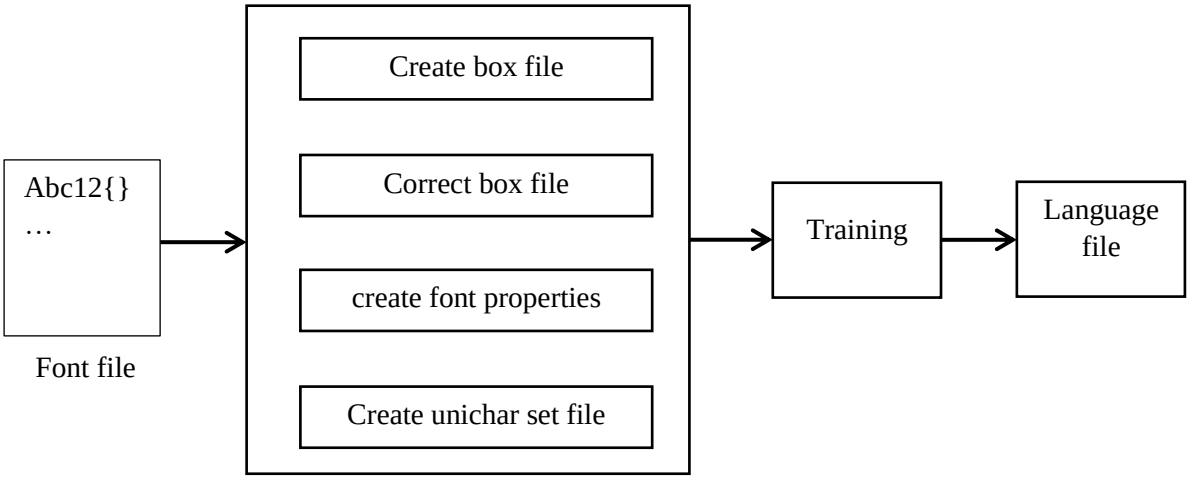


Figure 7 - Design of Language Training module

5.6 Character Recognition Module

The character recognition module does not have a separate and dedicated interface, just having a button called OCR for character recognition process after creating the language file.

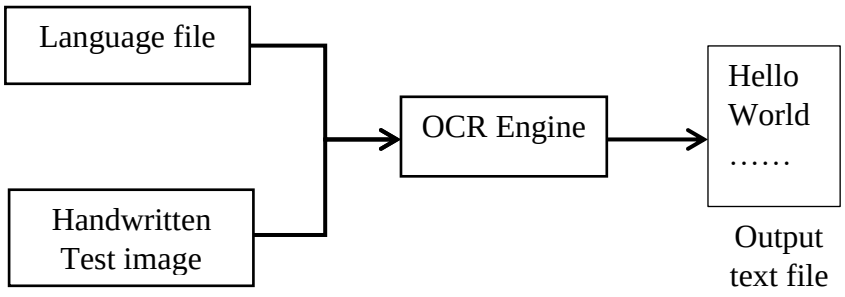


Figure 8 - character recognition module

5.7 Post processing and execution module

The post processing and the execution module have the separate interface with seven dedicated buttons to format the code, check the brackets, check for the correct Java syntaxes, correct the Java syntaxes, save and execute the recognized program.

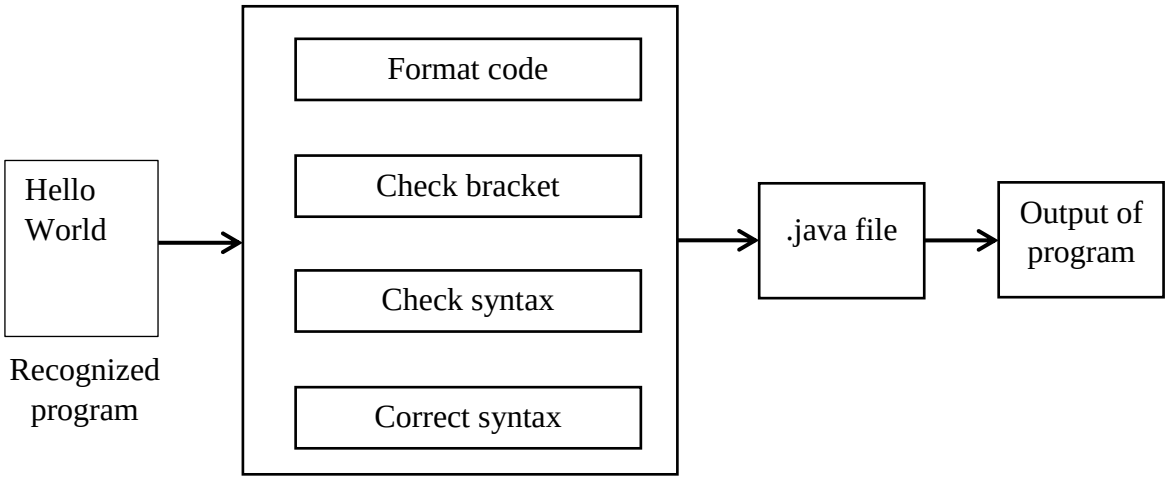


Figure 9 - Design of post processing Module

5.7.1 Format Code

As the first step of post processing, remove the unnecessary spaces between words and lines. Other than that ultility library are added to the program code.

5.7.2 Check Brackets

In this step, three methods have used to check three types of brackets, square brackets, curly brackets and parentheses. Using the implementation of stacks, checked whether all the open brackets are closed or not.

5.7.3 Check & Correct Syntax

Using the predefined set of Java syntax checked each and every word in the program, whether they are syntax or not. If it is a syntax, then check for the case sensitivity and corrected if it is in the incorrect case.

5.8 Summary

This chapter included the, extended design of the process and what the system does. It described several modules, namely user interface, preprocessing module, language training, character recognition, post processing and program execution. The relationships and links between above modules have described clearly. This chapter has discussed what the novel solution is. With a clear idea about the design of each module move to the next chapter, which describes how system has implemented as module wise and how entire solution get work.

Chapter 6

6 Implementation

6.1 Introduction

The previous chapter gave full detail of the design of preprocessing module, what system does the purpose of each and every module and link between those modules. This chapter describes the implementation of the each module of our novel approach with respect to the interface module, preprocessing module, language training module, character recognition module and the post processing module. It describes the implementation and how to build a prototype using the help of material such as some code segments, captured images of interfaces and main functions

6.2 Overall Solution

An overall solution has been implemented as a standalone Windows based application that can accessed by any client running on windows operating system. Workable prototype is primarily a C# based solution. Open source Tesseract libraries have used for the some character recognition parts.

6.3 Implementation of the interface module

A user-friendly front end interface as shown in Fig.10,Fig.11,Fig.12 have been implemented for the proposed handwritten character recognition system using menu based GUI (Graphical User Interface). There are three main user interfaces have been provided for the user operations, namely image preprocessor, language trainer, post processor and executor.

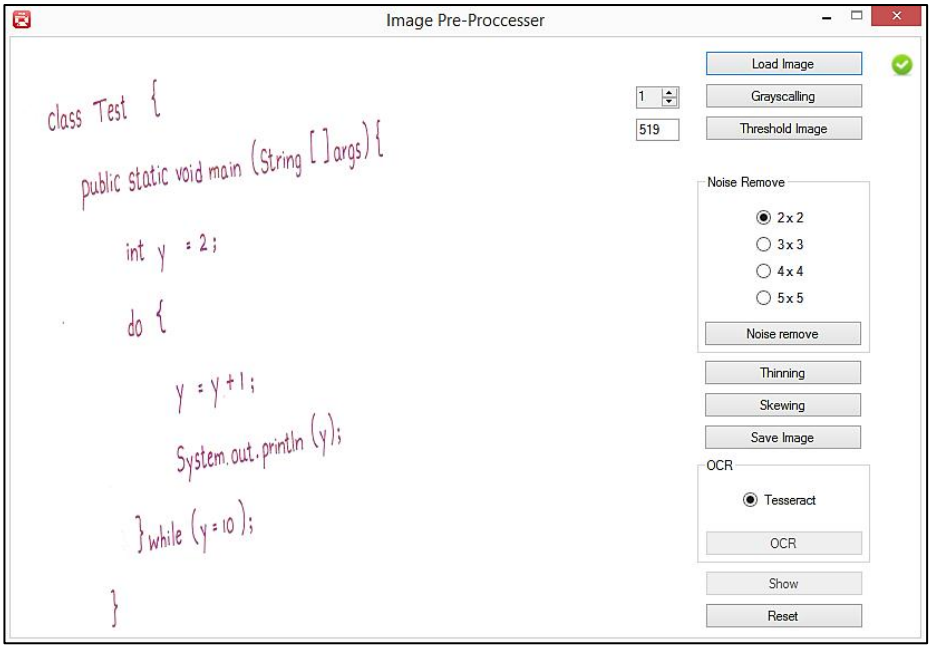


Figure 10 - User Interface for image acquisition and pre-processing

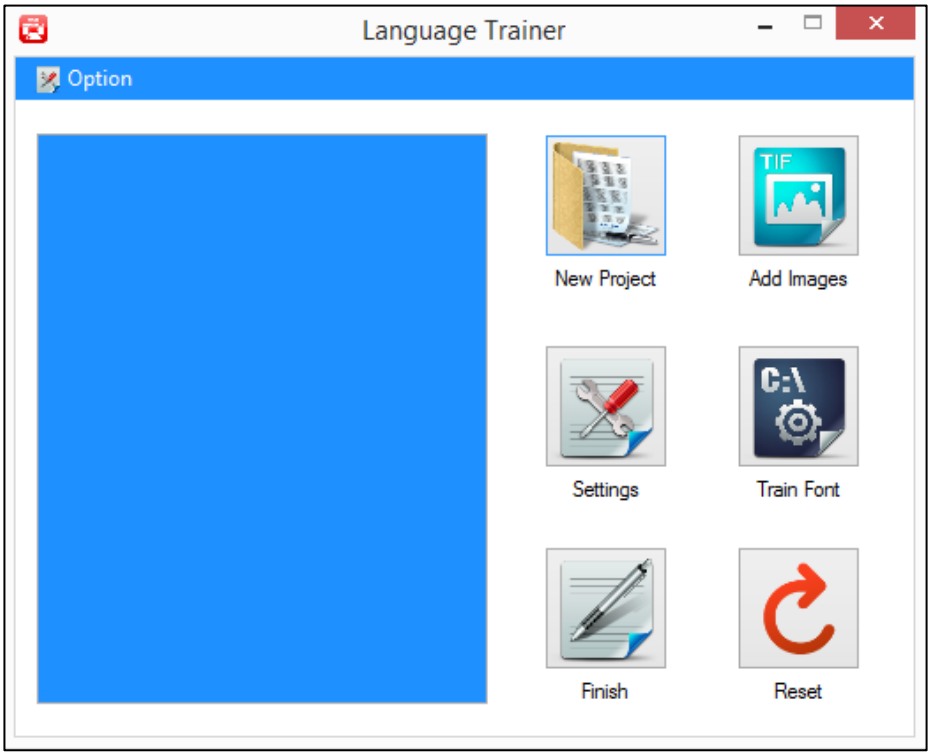


Figure 11 - User Interface for OCR Training

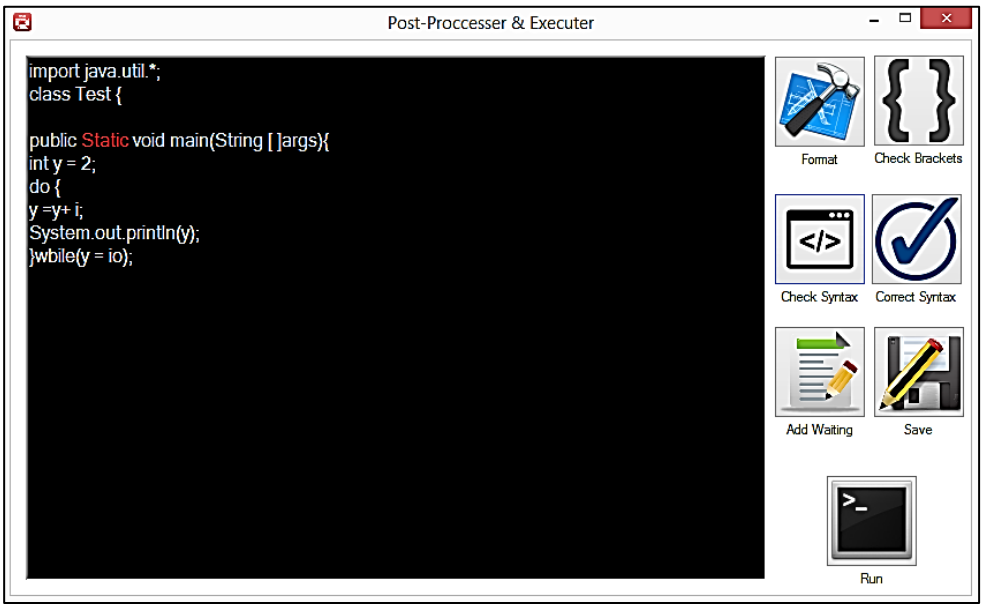


Figure 12 - User Interface for post-processing

6.4 Implementation of the preprocessing module

In the preprocessing, an image undergoes in five steps, gray scaling, thresholding, noise removal, thinning and skewing.

6.4.1 Gray Scaling

In photography and computing, a grayscale or grayscale digital image is an image in which the value of each pixel is a single sample, that is, it carries only intensity information. Images of this sort, also known as black-and-white, are composed exclusively of shades of gray, varying from black at the weakest intensity to white at the strongest. Therefore, as the first step of reprocessing, acquired image converted to black and white image.

For example, of gray scaled image, (see Appendix A)

6.4.2 Thresholding

Thresholding is a process of converting a grayscale input image to a bi-level image by using an optimal threshold. The purpose of thresholding is to extract those pixels from some image which represent an object. Otsu's global thresholding has used for threshold the gray scaled image.

For example, of theshold image, (see Appendix B)

6.4.3 Noise Removal

The noise, introduced by the optical scanning device or the writing instrument, causes disconnected line segments, bumps and gaps in lines, filled loops, etc. The distortion including local variations, rounding of corners, dilation and erosion, is also a problem. Prior to the character recognition, it is necessary to eliminate these imperfections. For the application three types of optional median filters (3x3, 4x4, and 5x5) were used.

6.4.4 Thinning

Thinning is an image preprocessing operation performed to make the image crisper by reducing the binary-valued image regions to lines that approximate the skeletons of the region. Thinning cleans the image so that only a reduced amount of data needs to be processed in the next image processing stage. Shape analysis could be done easily.

For example, of Thinning image, (see Appendix C)

6.4.5 Skewing

Handwritten document may originally be skewed or skewness may introduce in the document scanning process. This effect is unintentional in many real cases, and it should be eliminated because it dramatically reduces the accuracy of the subsequent processes, such as segmentation and classification. Skewed lines are made horizontal by calculating the skew angle and making a proper correction in the raw image.

For example, of skewed image, (see Appendix D)

6.5 Implementation of Training Module

As mentioned in the process, there are two main steps in language training.

- Create test language using 'boxing'
- Train for the new test language

In order to create the test language, image should be saved in '.tif' format. Using that .tif image box file generated and input that into the working directory manually using the following command.

tesseract ntl.new_test_language.exp0.tif ntl.new_test_language.exp0 batch.nochop makebox

Training language should be denoted by three letter acronym. Therefore the box file's and the .tif image file's naming convention should be:

Language name: new_test_language

Acronym: ntl

Box file: ntl.new_test_language.exp0.box

Image file: ntl.new_test_language.exp0.tif

Referred box file created in the working directory checked for the correct boxing. For the box file checking and the correction, one of the available box editors was used. Fig.13 shows an one of the box file editor.

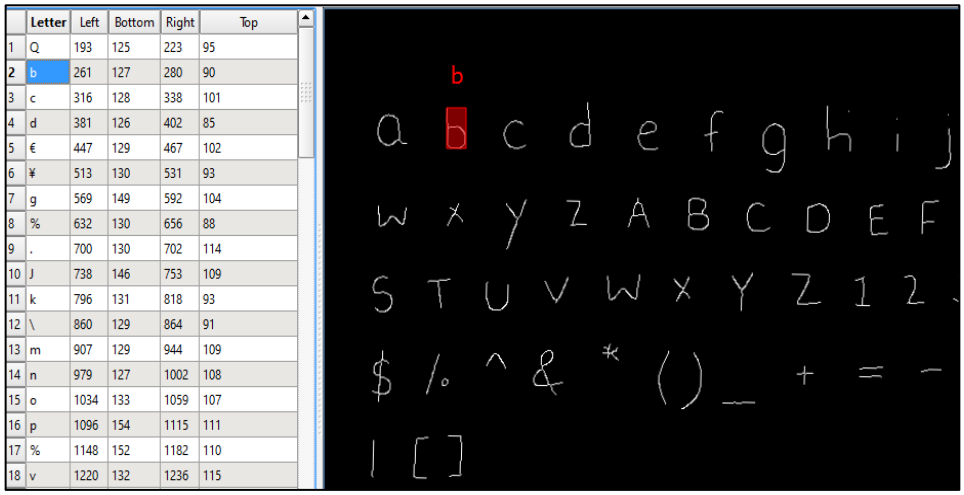


Figure 13 - Box File Editor

After the creation of box file, font properties generated for the new language.

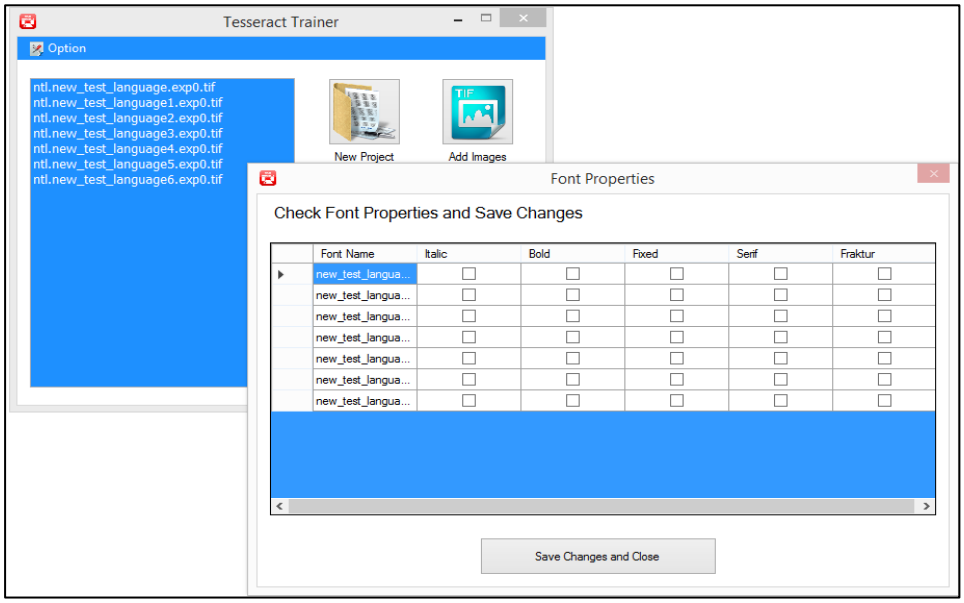


Figure 14 –Gearating font properties

As training is the second main step ‘unicharset’ file and the ‘master shape table’ created and passed those two files into the Tesseract ‘mtraining’ and the ‘cntrainig, processes. In the training process ‘inttemp’, ‘normproto’, ‘pffmtble’ and the ‘shapetable’ need to create with respect to the language name.

- ntl.inttemp
- ntl. normproto
- ntl. pffmtble
- ntl. shapetable

Finally, these four files combined in order to complete the language training. The new language file can be found in the working directory.

6.6 Implementation of Post processing and Execution module

In post processing, the system automatically formats the recognized program as the first step. Then checked for the brackets of the program. If there is any missing bracket it notified the incomplete bracket in red color.

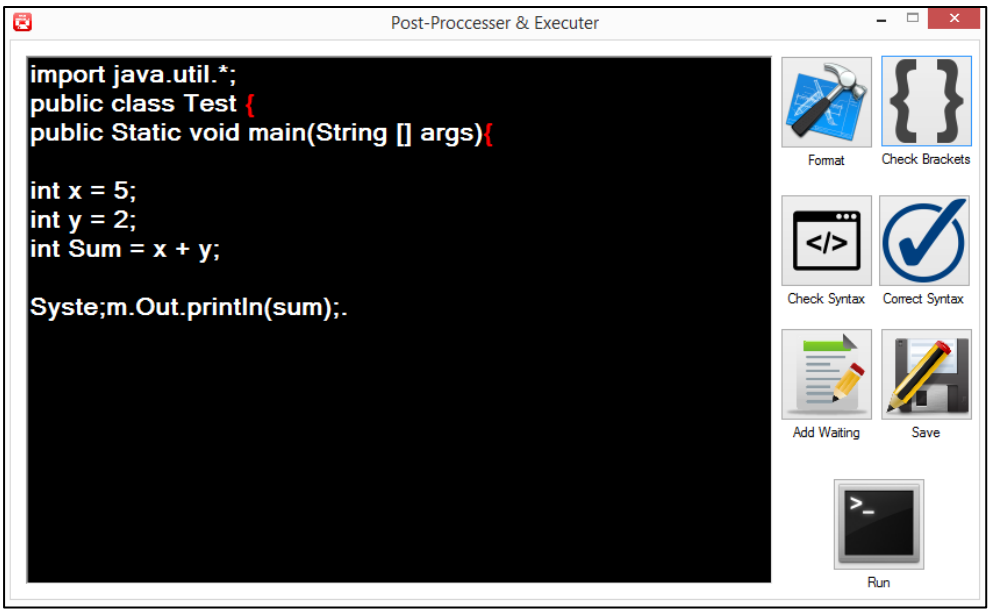


Figure 15 – Check bracket in post-proccessing

After complete the bracket checking check for the incorrect Java syntaxes and correct the erroneous identification of syntaxes. Error syntaxes also notified in red color.

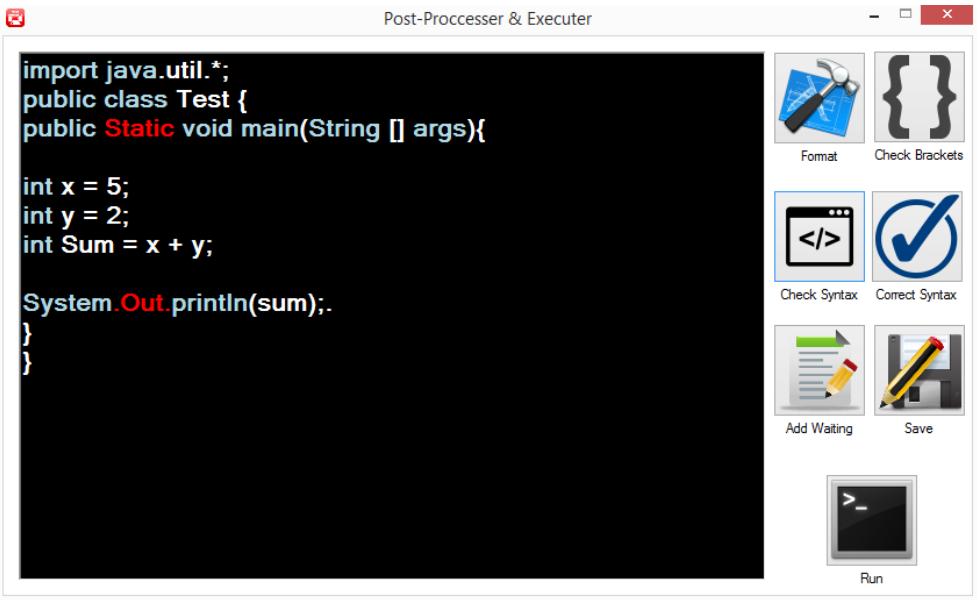


Figure 16 - Check Syntax in post-proccessing

Finally corrected program can be compiled and execute for the expected result.

6.7 Summary

This chapter described how the new system has developed and what are the main implementation modules and described the each and every module separately With clear Idea about the system implementation, move to the next chapter, which describes the evaluation of the novel system in order to present how much accurate the proposed approach is.

Chapter 7

7 Evaluation

7.1 Introduction

The previous chapter described the implementation of the system. This chapter described the evaluation of experimental results. Evaluation has done with respect to the accuracy of handwriting identification. Accuracy evaluated as a percentage of correctly identified characters or the programming syntaxes. Accuracy presented as a graphical representation with a comparison to the image before the train and the image after the train for recognition.

7.2 Participants

There is no any special group of participants involved in the evaluation. The participants were the several persons who wrote the several program snippets.

7.3 Testing Environment

As the testing environment, standalone offline application environment was used. There was no use of special testing tools even.

7.4 Data collection

As the input data, images of the handwritten program and the image of sample characters of the same user were used. The sample character sheet includes upper and lower case letters, 0-9 numbers, all the wild card characters in computer keyboard. For the evaluation result five image samples were used.

7.5 Data Analysis

Data was analyzed based on randomly selected five images and same image of a program of different users. The randomly selected images mainly considered the correct recognition of character count and the correct recognition of the case sensitivity. Accuracy of recognition before the training and after the training can be shown as below Fig.17

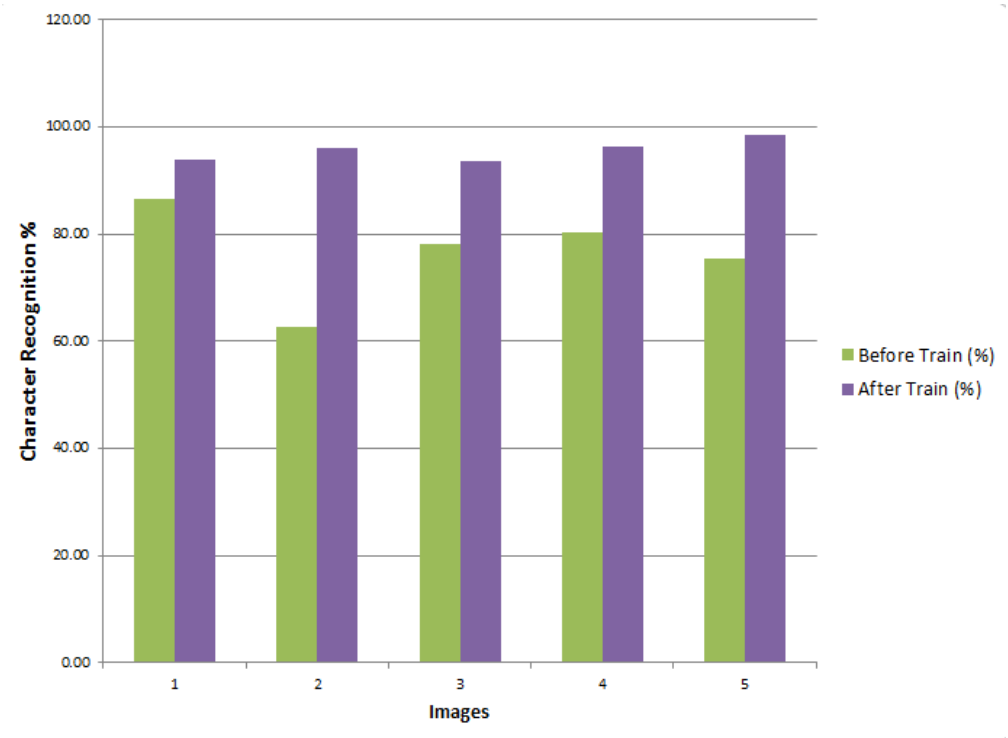


Figure 17 – Character recognition parentage

The above figure clearly shows that the character recognition rate is increasing after the training process for five different images. The character recognition rate before the training is always lower than the character recognition rate of after the training. The average difference of the character recognition rate for the evaluated images is shown in Fig.18

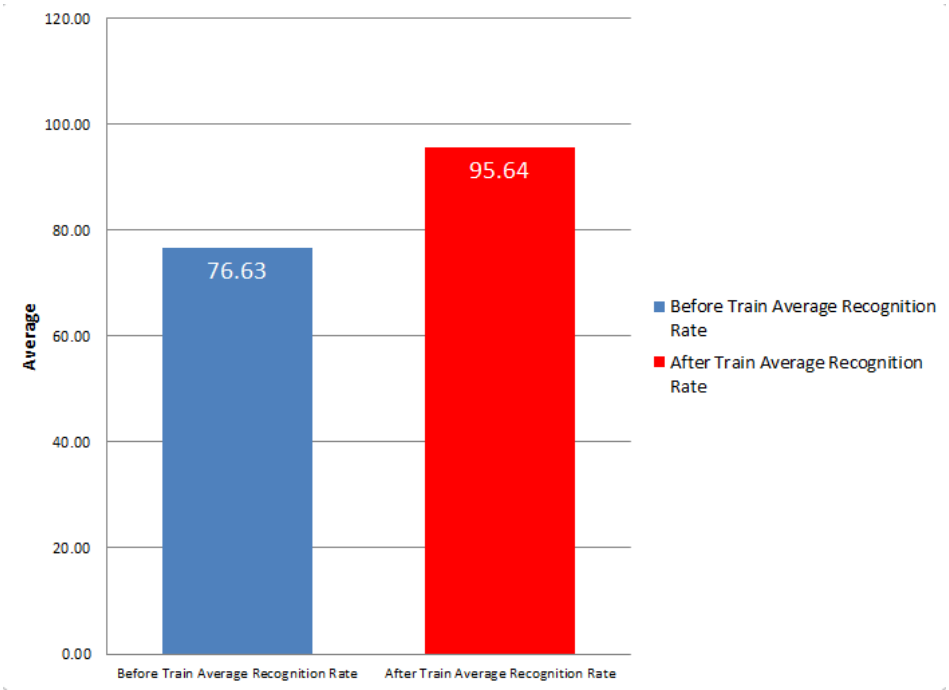


Figure 18 – Average character recognition rate

After that recognition rate evaluated of the same image of different users. The recognition rate and the average of recognition is shown in the Fig.19 and 20.

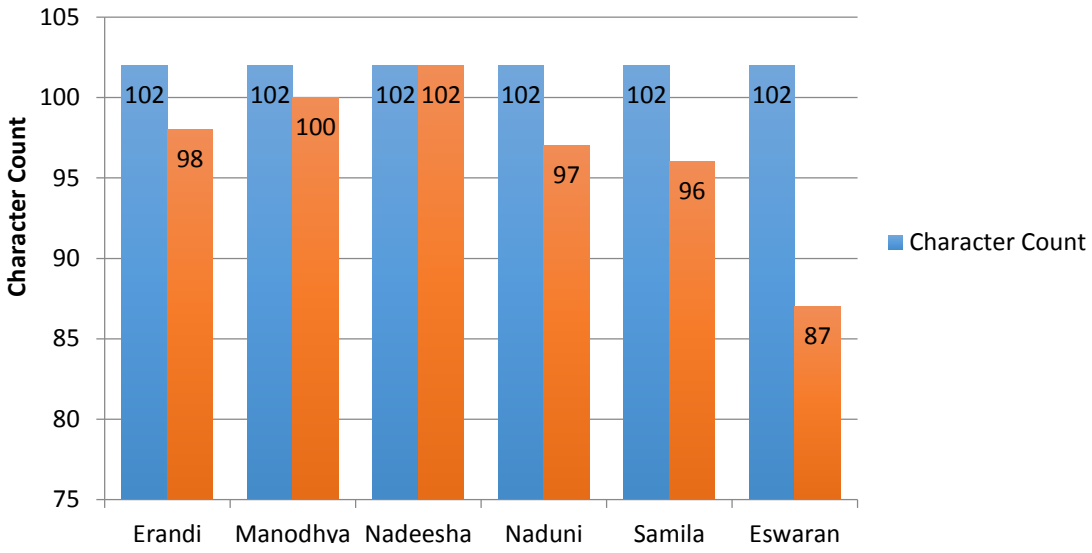


Figure 19 – Recognized character count for different users

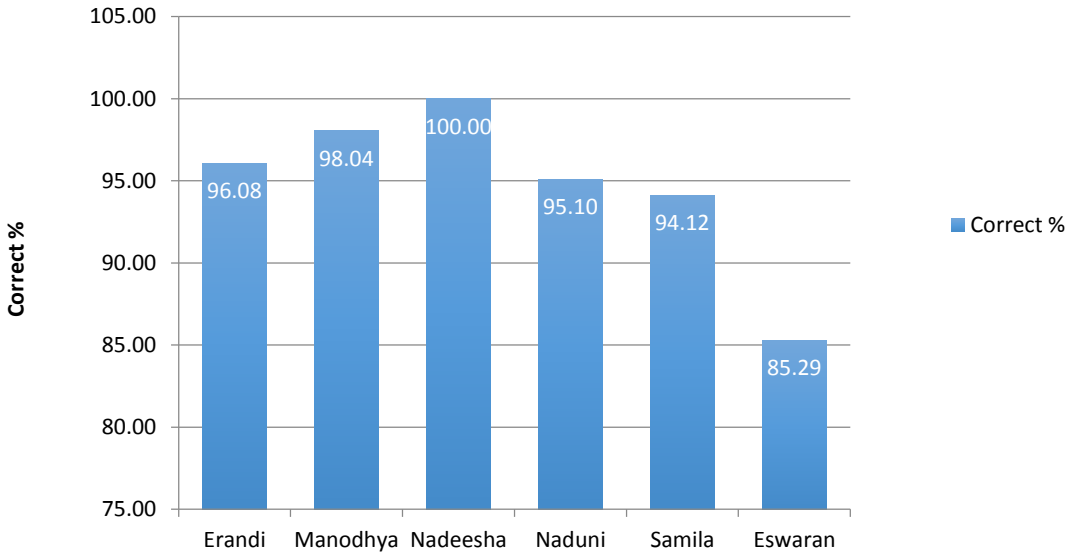


Figure 20- Average recognition rate for different users

Then the character recognition rate is evaluated for the case sensitivity. Here also case sensitivity recognition rate is always higher for the trained images with compared to the untrained images. The average correct case sensitivity for untrained images is 76.63% and the average correct case sensitivity for the trained images is 95.64%. It shows that the average correct case sensitivity is increased after the training has done

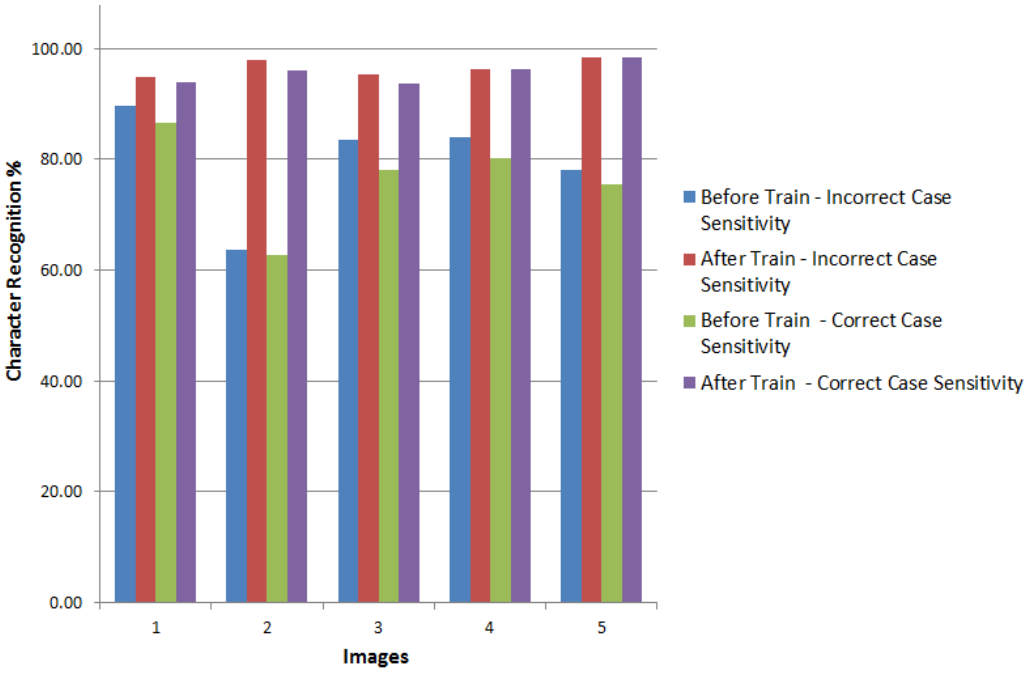


Figure 21 – Character recognition on case sensitivity

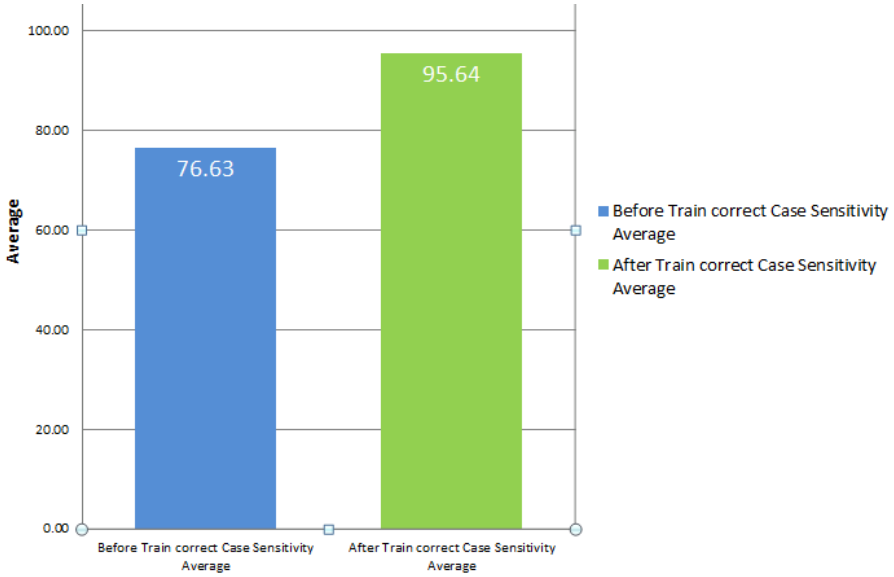


Figure 22 – Average character recognition on case sensitivity

Other than the before and after recognition rate and the case sensitivity, identified character count also evaluated. The evaluation clearly shows that there is a significant difference in reading character count after the training. But it has a slight difference with the actual character count of the image. An average identified character count before the training is 88 and an average character count after the training is 110.8, while an actual average character count of the images is 115.4.

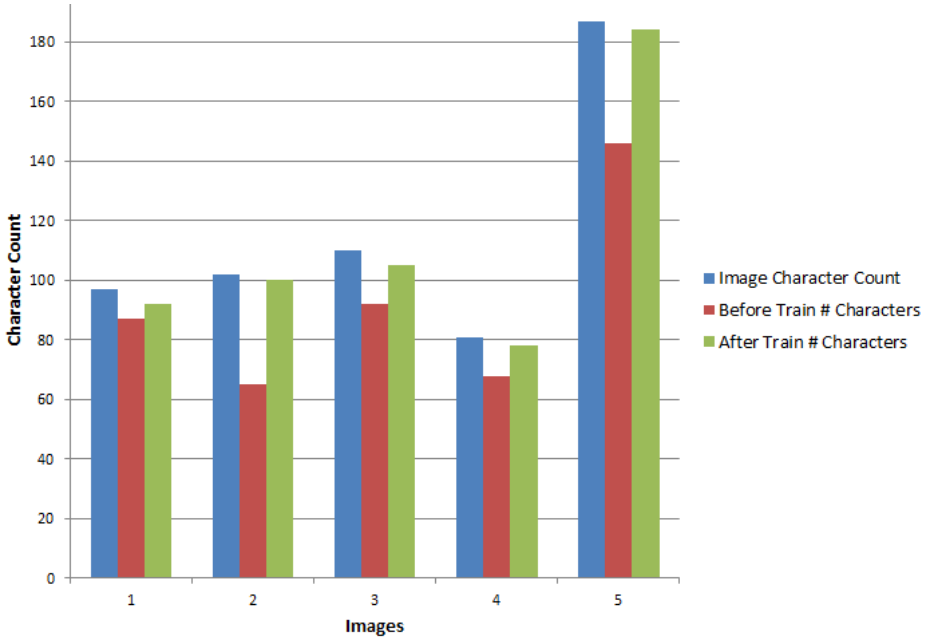


Figure 23 – Character recognition count

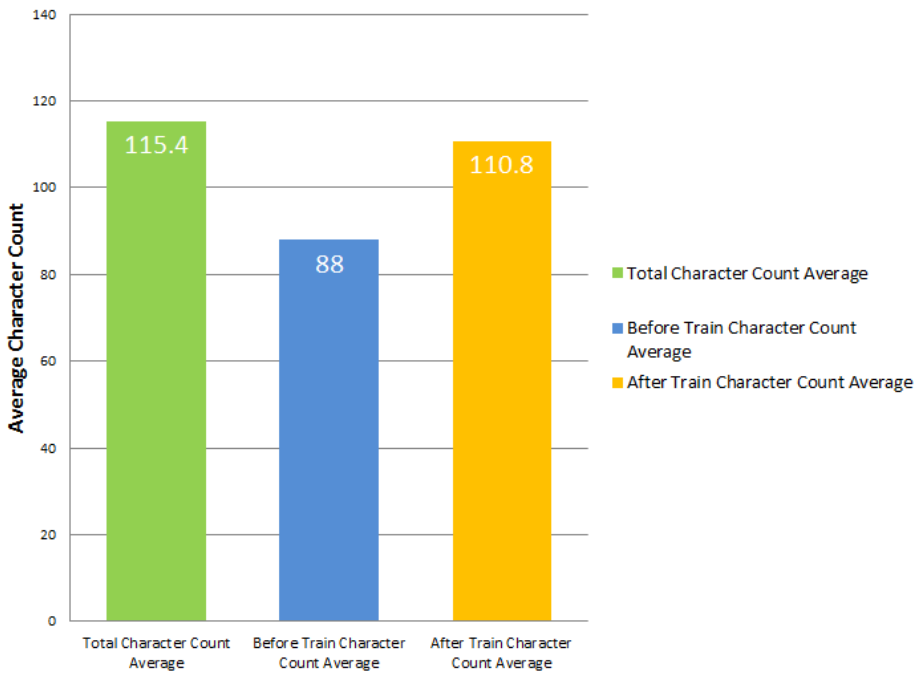


Figure 24 – Average character recognition count

7.6 Summary

This chapter has presented an evaluation of the novel handwriting recognition system. For the evaluation mainly focused on the accuracy of the recognition. It is evaluated against the character count, correct case sensitivity and the training. After critically analyzed the rate of handwritten recognition move to the next chapter, which describes the future works and final conclusion of the works done so far.

Chapter 8

8 Conclusion

8.1 Introduction

Chapter7 has presented an evaluation of research processes. This chapter of thesis, We have described the conclusion of our research findings, according to the results of our critical evaluation. To this point We have collected materials, dataset and results to check our hypothesis. In this chapter We will present the critical review of our research works against the experimental results. The We will write the conclusions of our research works

8.2 Overview of the research

There has been much research and the attempts of character identification, but most of them are limited only to the printed character identification or to the just identifying of handwritten of the English alphabet or the 0-9 digits. Therefore, it could be revealed that there are no any service or the facility to automatically read the handwritten computer program.

In this research, it is used that the blend of image processing and the training methodology using the Tesseract OCR engine in order to read the handwritten program. According to the evaluation result, it shows that the hypothesis is established and higher accuracy level of recognition has been achieved using the blend of image processing and the training methodology.

8.3 Major Findings

The evaluation has been done based on the number of characters in the input image, case sensitivity of the characters and the given training process. According to the evaluation, training and the preprocessing are the very critical phases in the recognition process. Accuracy always increased after the preprocessing and the

training. The recognition rate, evaluated with the training and the case sensitivity statistically shows that how the performance differentiate with the training and the preprocessing of the image. According to that the table shows how the rate of recognition has been increased. Therefore, It can be concluded that, higher accuracy of recognition (more than 95%) can be gained through the proper image preprocessing and the correct OCR training methodology. Overall statistic included for further references in Appendix E.

	Before Traiening	After Training
Average Recognition Rate (With Case Sensitive)	76.63456749	95.64424924
Average Recognition Rate (Without Case Sensitive)	79.81561182	96.60622804

Table 1 - Average recognition rate

	Original Image	Before Train	After Train
Average Character Count	115.4	88	110.8

Table 2 - Average Character Count

8.4 Future work

It is evident that the hypothesis has established and better to suggest some future works to the other researchers. One possible future development is an implementation of a better training algorithm for better accuracy. An another future solution is to generate the computer program for the pseudocode also will be a much interesting area of this type of research.

8.5 Summary

This chapter has given an overview of the total solution and depth discussion about the achievements. The paper included all the chapters together and discussed

methodology, experiments and evaluation done in the research process. Finally It is shown with evidence, the newly introduced system can successfully solve the identified research gap from the literature and the hypothesis has been successfully established.

References

[1] E. Borovikov, “A survey of modern optical character recognition techniques,” *ArXiv Prepr. ArXiv14124183*, 2014.

[2] A. Singh, K. Bacchuwar, and A. Bhasin, “A survey of OCR applications,” *Int. J. Mach. Learn. Comput.*, vol. 2, no. 3, p. 314, 2012.

[3] L. V. Rasmussen, P. L. Peissig, C. A. McCarty, and J. Starren, “Development of an optical character recognition pipeline for handwritten form fields from an electronic health record,” *J. Am. Med. Inform. Assoc. JAMIA*, vol. 19, no. e1, pp. e90–e95, Jun. 2012.

[4] S. Carbonnel and E. Anquetil, “Lexical post-processing optimization for handwritten word recognition,” in *Document Analysis and Recognition, 2003. Proceedings. Seventh International Conference on*, 2003, pp. 477–481.

[5] J. Jin, X. Han, and Q. Wang, “Mathematical Formulas Extraction.,” in *ICDAR*, 2003, pp. 1138–1141.

[6] J. R. Bruce, “Mathematical Expression Detection and Segmentation in Document Images,” Virginia Tech, 2014.

[7] R. J. Fateman, “How to find mathematics on a scanned page,” in *Proc. SPIE*, 1999, vol. 3967, pp. 98–109.

[8] G. Nagy, T. A. Nartker, and S. V. Rice, “Optical character recognition: An illustrated guide to the frontier,” in *Electronic Imaging*, 1999, pp. 58–69.

[9] P. M. Kamble and R. S. Hegadi, “Handwritten Marathi Character Recognition Using R-HOG Feature,” *Procedia Comput. Sci.*, vol. 45, pp. 266–274, 2015.

[10] A. Choudhary, R. Rishi, and S. Ahlawat, “Off-line Handwritten Character Recognition Using Features Extracted from Binarization Technique,” *AASRI Procedia*, vol. 4, pp. 306–312, 2013.

[11] A. Bal and R. Saha, “An Improved Method for Handwritten Document Analysis Using Segmentation, Baseline Recognition and Writing Pressure Detection,” *Procedia Comput. Sci.*, vol. 93, pp. 403–415, 2016.

[12] Y.-N. Chang, S. Furber, and S. Welbourne, “Modelling normal and impaired letter recognition: Implications for understanding pure alexic reading,” *Neuropsychologia*, vol. 50, no. 12, pp. 2773–2788, Oct. 2012.

[13] M. Ryan and N. Hanafiah, “An Examination of Character Recognition on ID card using Template Matching Approach,” *Procedia Comput. Sci.*, vol. 59, pp. 520–529, Jan. 2015.

[14] W. A. Barrett and H. E. Nielson, “Consensus-based table form recognition,” 2003.

[15] A. Kornai, C. Scott, and others, “Recognition of cursive writing on personal checks,” 1996.

[16] X. Xiaobing, W. Xiaoxu, W. Jianping, Z. Chenghui, and Q. Yuping, “Recognition Research of Offline-handwritten Chinese Character Based on Biomimetic Pattern,” *Procedia Eng.*, vol. 15, pp. 5116–5120, 2011.

[17] S. Celar, Z. Stojkic, Z. Seremet, Z. Marusic, and D. Zelenika, “Classification of Test Documents Based on Handwritten Student ID’s Characteristics,” *Procedia Eng.*, vol. 100, pp. 782–790, 2015.

[18] E. Tapia and R. Rojas, “Recognition of on-line handwritten mathematical expressions using a minimum spanning tree construction and symbol dominance,” in *International Workshop on Graphics Recognition*, 2003, pp. 329–340.

[19] J. F. Pitrelli and M. P. Perrone, “Confidence-scoring post-processing for off-line handwritten-character recognition verification,” in *Document Analysis and Recognition, 2003. Proceedings. Seventh International Conference on*, 2003, pp. 278–282.

[20] I. Bhattacharya, P. Ghosh, and S. Biswas, “Offline Signature Verification Using Pixel Matching Technique,” *Procedia Technol.*, vol. 10, pp. 970–977, 2013.

[21] M. Cheriet, “Shock filters for character image enhancement and peeling,” 2003, vol. 1, pp. 1247–1251.

[22] Y. Zheng, H. Li, and D. Doermann, “Text identification in noisy document images using Markov random model,” in *Document Analysis and Recognition, 2003. Proceedings. Seventh International Conference on*, 2003, pp. 599–603.

[23] M.-C. Jung, Y.-C. Shin, and S. N. Srihari, “Multifont classification using typographical attributes,” in *Document Analysis and Recognition, 1999. ICDAR’99. Proceedings of the Fifth International Conference on*, 1999, pp. 353–356.

[24] A. Kornai, “An experimental HMM-based postal ocr system,” in *Acoustics, Speech, and Signal Processing, 1997. ICASSP-97., 1997 IEEE International Conference on*, 1997, vol. 4, pp. 3177–3180.

[25] M. Mirmehdi, P. Clark, and J. Lam, “Extracting low resolution text with an active camera for OCR,” in *Proceedings of the IX Spanish Symposium on Pattern Recognition and Image Processing*, 2001, pp. 43–48.

[26] T. Abu-Ain, S. N. H. S. Abdullah, B. Bataineh, W. Abu-Ain, and K. Omar, “Text Normalization Framework for Handwritten Cursive Languages by Detection and Straightness the Writing Baseline,” *Procedia Technol.*, vol. 11, pp. 666–671, 2013.

[27] W. Du, “Code Runner: Solution for Recognition and Execution of Handwritten Code.”

[28] M. M. M. Fahmy, “Online handwritten signature verification system based on DWT features extraction and neural network classification,” *Ain Shams Eng. J.*, vol. 1, no. 1, pp. 59–70, Sep. 2010.

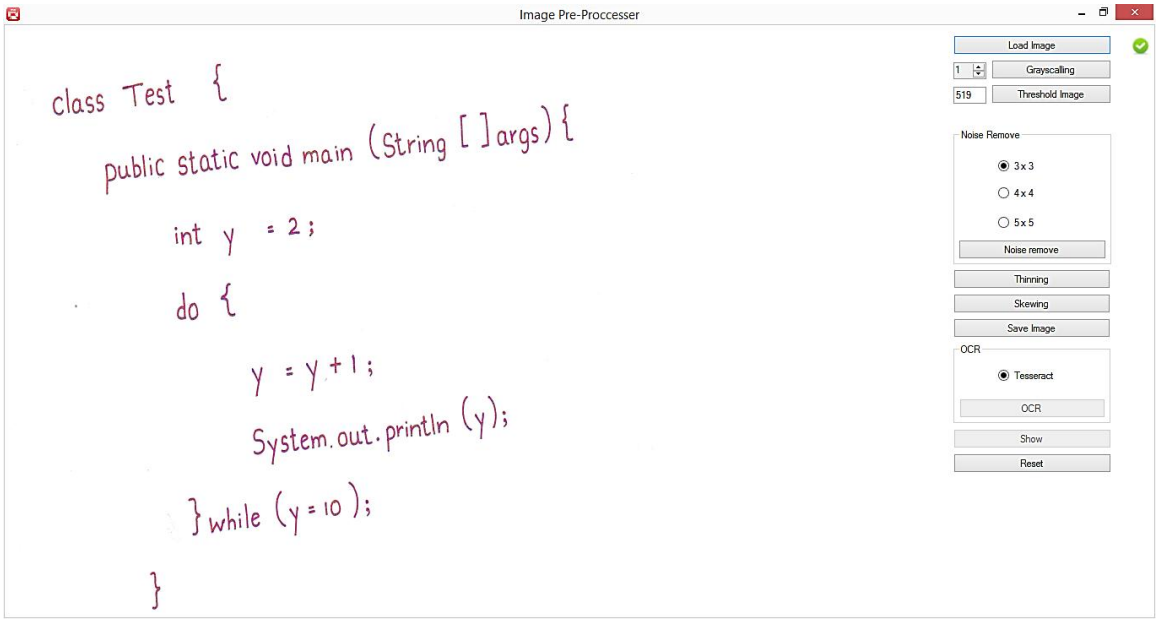
[29] S. Malakar, R. K. Das, R. Sarkar, S. Basu, and M. Nasipuri, “Handwritten and Printed Word Identification Using Gray-scale Feature Vector and Decision Tree Classifier,” *Procedia Technol.*, vol. 10, pp. 831–839, 2013.

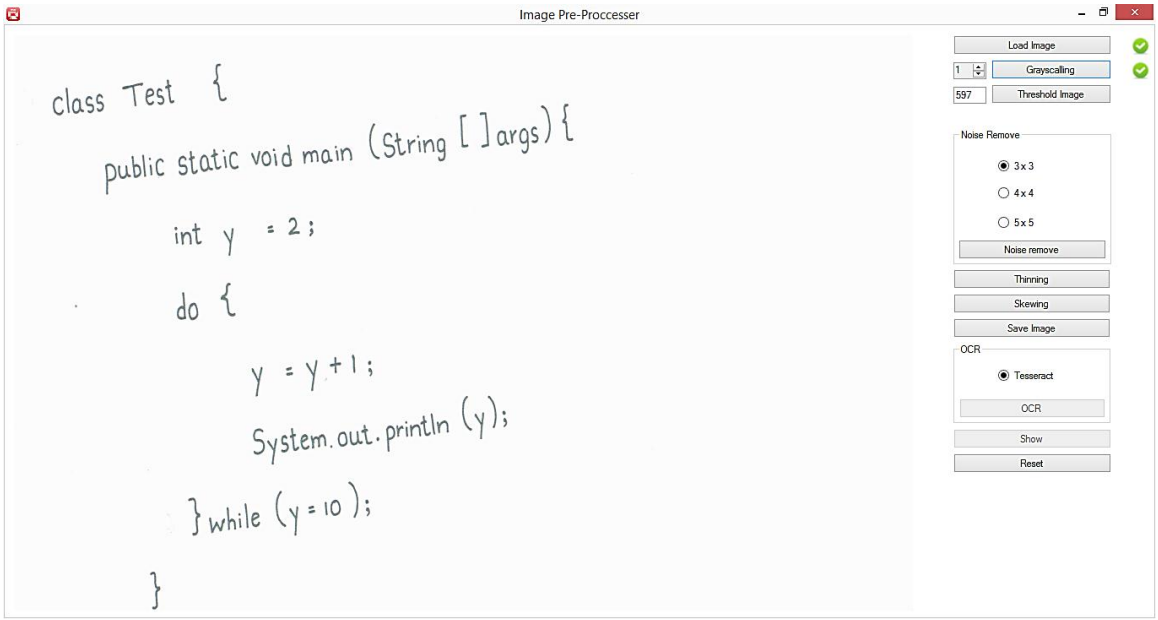
[30] N. Samadiani and H. Hassanpour, “A neural network-based approach for recognizing multi-font printed English characters,” *J. Electr. Syst. Inf. Technol.*, vol. 2, no. 2, pp. 207–218, Sep. 2015.

[31] B. Borjigin, “An Overview of the Tesseract OCR Engine.”

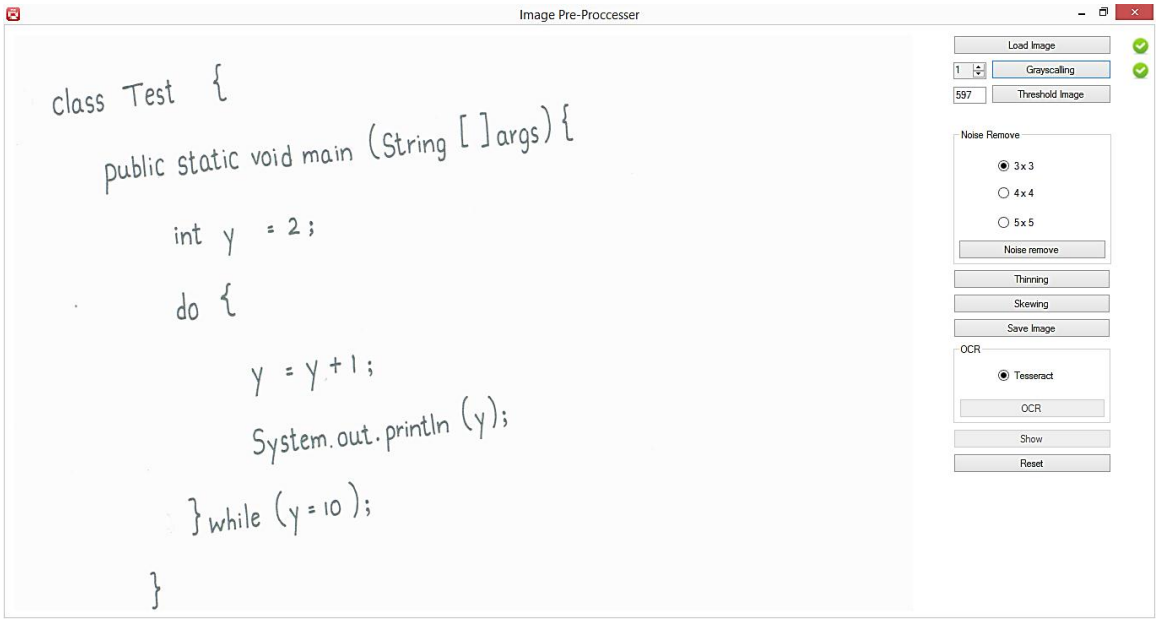
Appendix

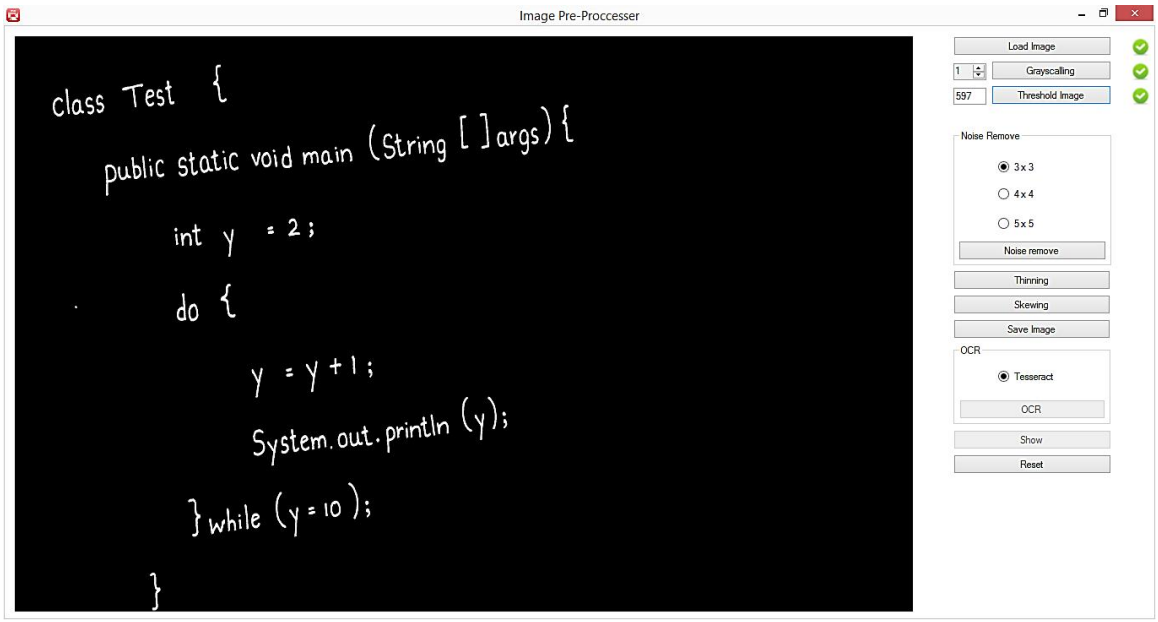
Appendix A - Sample for gray scaled image



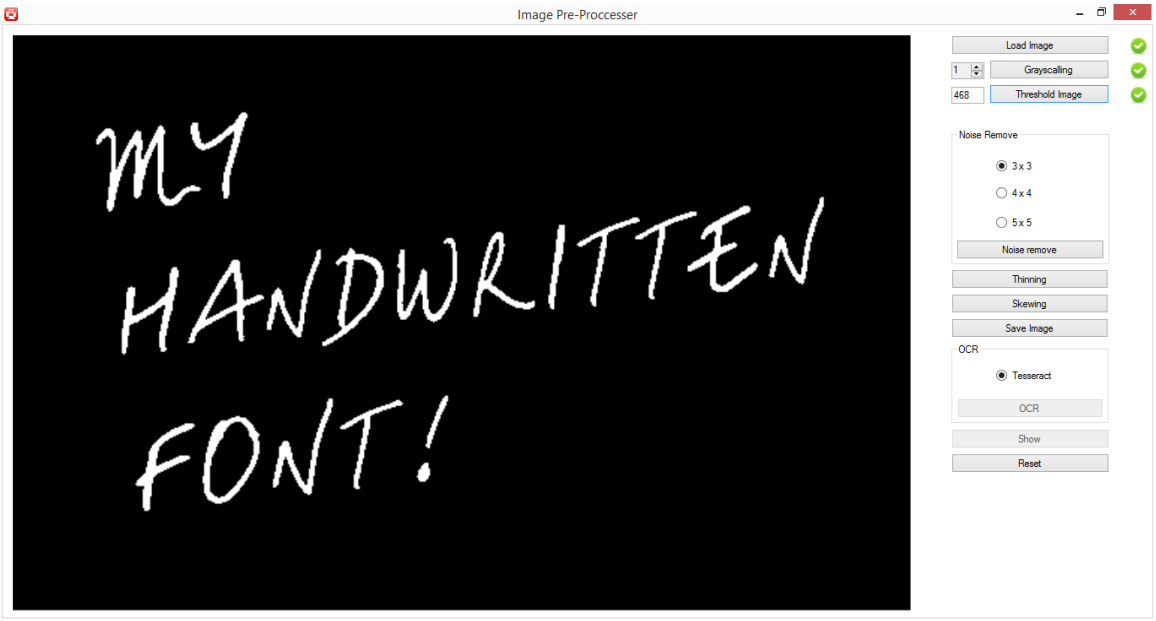


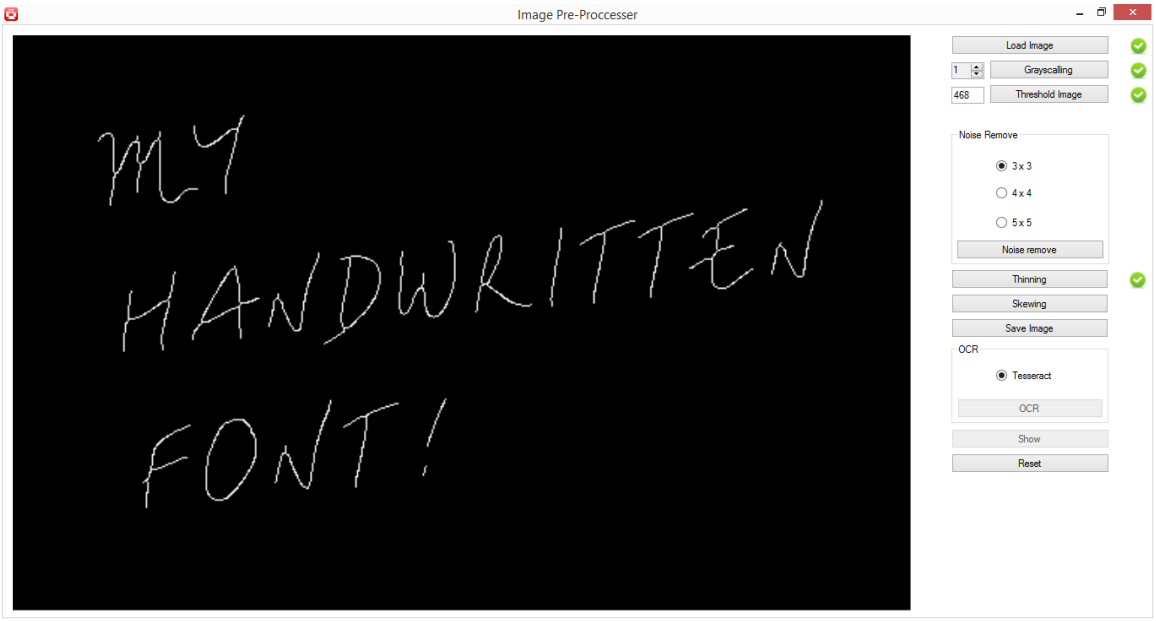
Appendix B - Sample for theshold image



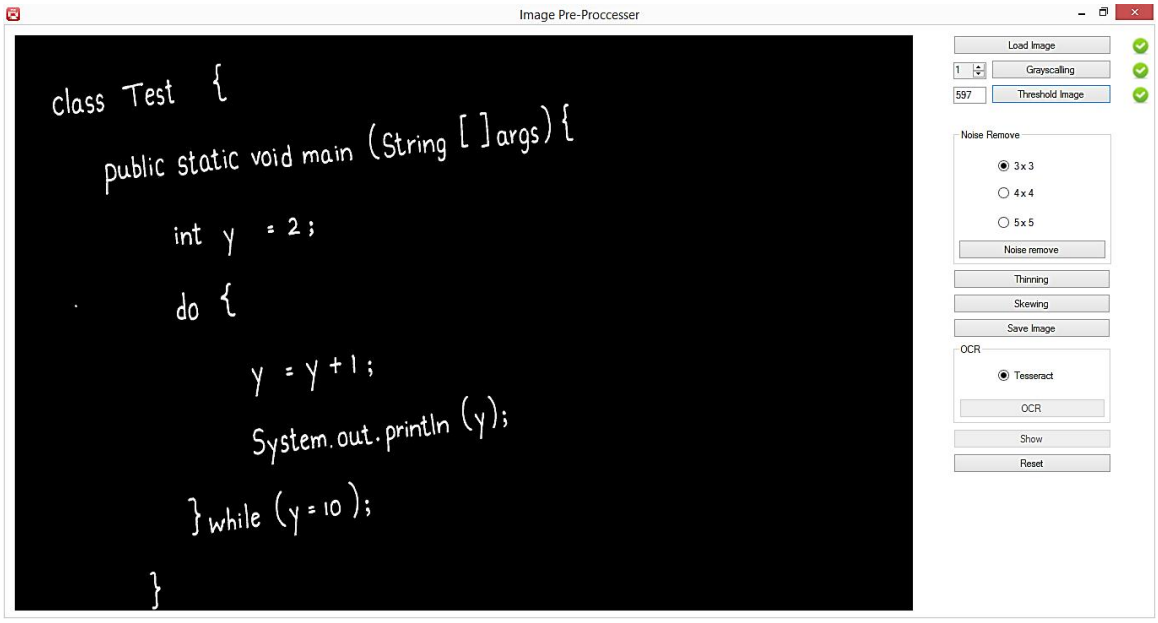


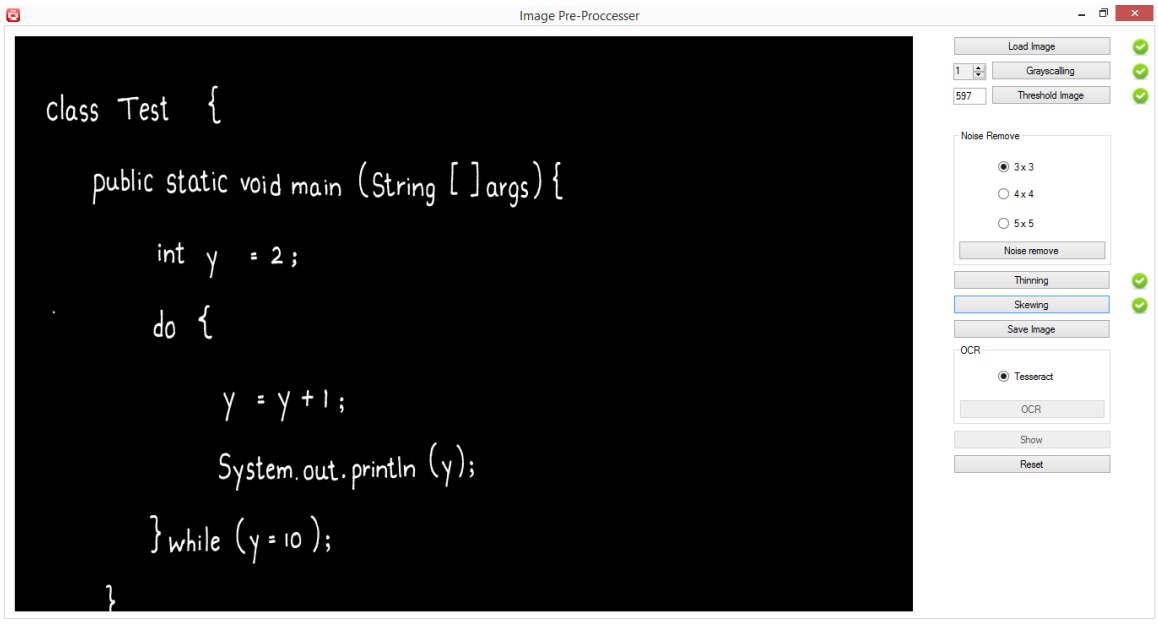
Appendix C - Sample for Thinning image





Appendix D - Sample for skewed image





Appendix E - Overall statistic

Image	Character count	Before Train				After Train			
		Incorrect Case Sensitivity		Case Sensitive		Incorrect Case Sensitivity		Case Sensitivity	
		Characters	%	Characters	%	Characters	%	Characters	%
1	97	87	89.69	84	86.60	92	94.85	91	93.81
2	102	65	63.73	64	62.75	100	98.04	98	96.08
3	110	92	83.64	86	78.18	105	95.45	103	93.64
4	81	68	83.95	65	80.25	78	96.30	78	96.30
5	187	146	78.07	141	75.40	184	98.40	184	98.40
SUM	577	458	399.08	440	383.17	559	483.03	554	478.22
AVG	115.4	91.6	79.82	88	76.63	111.8	96.61	110.8	95.64