Segmentation of Overlapping Sinhala Handwritten Characters

Kalhari Shanthika Ahangama Walawage

169339A

Faculty of Information Technology

University of Moratuwa

February 2019

Segmentation of Overlapping Sinhala Handwritten Characters

Kalhari Shanthika Ahangama Walawage

169339A

Dissertation submitted to the Faculty of Information Technology, University of Moratuwa, Sri Lanka for partial fulfillment of the requirements of Master of Science in Information Technology

February 2019

Declaration

I declare that this thesis is my own work and has not been submitted in any form for another degree or diploma at any university or other institution of tertiary education. Information derived from the published or unpublished work of others has been acknowledged in the text and a list of references is given.

Name of Student: K.S.A. Walawage

Signature of Student:

Date:

Supervised by

Name of Supervisor: Dr. L. Ranathunga

Signature of Supervisor:

Date:

Acknowledgements

I would like to express my sincere gratitude to my supervisor Dr L. Ranathunga, Senior Lecturer, Faculty of Information Technology, University of Moratuwa for his support, patience, motivation, and immense knowledge. His guidance helped me throughout my MSc. research and writing of this thesis.

I'm grateful to my parents, Mr. A.W.B. Chandrasena and Ms. B.D.G. Liyanage for their enormous support given to me at various points in time during this research.

I thanks to my brother and sister for supporting me in any time of my research work.

Finally, I thanks to friends who motivated and supported me.

Abstract

Sinhala is the official and national language of Sri Lanka. Seventeen million people of Sri Lanka use Sinhala language to their day to day works. Most of the researches have been done to Sinhala printed character recognition with high accuracy. Nowadays, Sinhala handwritten character recognition is popular research in Sri Lanka. It is not like printed character segmentation; shape of the same type of handwritten character can be changed in different times. Therefore, characters will be overlapped or touched with each other. Handwritten character segmentation is more important to increase the accuracy of the character recognition. Currently there is lack of high accuracy finding to segment overlapping and touching Sinhala handwritten characters. The proposed methodology has six main sections. They are image acquisition, preprocessing, segmentation, classification, feature extraction and recognition. Collected image was loaded to the system and preprocessed it. Preprocess section is included noise removing, thresholding etc. After that, text line segmentation was done using horizontal projection profile. Mainly this research was introduced a connected pixel labeling method to segmentation of overlapping characters and peak and valley point identification method to segmentation of touching characters. According to tested result, connected pixel labelling method has 97% accuracy and peak and valley identification method has 72% accuracy.

Keywords - Sinhala, Overlapping, Touching, Segmentation, Connected Pixels Labeling, Peak and Valley Point Identification

Table of Contents

List of Figures

List of Tables

Chapter 1
Introduction

1.1 Chapter Introduction

This chapter has three sections. First section briefly describes introduction about background and motivation for the research. Second section describes Aim and Objectives of the research. Last section describes my research solution.

1.2 Background and Motivation

Sinhala Language is native language of people of Sinhalese. These people are the largest ethnic group in Sri Lanka. There are nearly 17 million Sinhalese people. Also 2 million people are speaking Sinhala as second language in Sri Lanka. Sinhalese belongs to Indo-Aryan branch and Indo-European language family [1].

In Sri Lanka, huge amount of official Sinhala documents such as forms, letters, cheques and faxes are handwritten. Many government documents are still written in Sinhala. Nowadays most of the printed documents are created manually typing by a person. It wastes the more times of the offices; Automatically Sinhala handwriting recognition is highly essential to creating a printed document by machine [2].

In the last decade, recognition of Sinhala characters using a computer is a popular problem in the research field. Some of the researches have been recognized machine-printed Sinhala characters [3], [4]. Sinhala handwritten character recognition is some difficult task than printed character recognition. Recently some percentage of handwritten characters were recognized in certain researches [2], [5], [6].

Research problem is segmentation of overlapping and touching Sinhala handwritten characters to improve the accuracy of recognition of Sinhala handwritten characters.

Some previous research has been done line, word and basic character segmentation [2], [5], [7]. They were not perfectly segmented the overlapping and touching Sinhala handwritten characters. Handwritten characters segmentation of other languages was done with some accuracy. Conjunct and overlapping characters in Devanagari script on Hindi language were segmented using cluster detection technique [8]. Offline cursive handwritten Tamil character segmentation has been done and Hidden Markov Models was used to recognize the characters [9].

1.3 Aim and Objectives

1.3.1 Aim

This research is mainly focusing on segmentation of overlapping and touching Sinhala handwritten characters.

1.3.2 Objectives

- A method of segmentation of Overlapping Sinhala handwritten characters was found and implemented successfully.
- A method of segmentation of Touching Sinhala handwritten characters was found and implemented successfully.

1.4 Solution

In this research, image acquisition, preprocessing and segmentation are the main steps. First handwritten document must be collected. An image of that document was acquired from a scanner or digital camera. That image was preprocessed using noise removing and binarization. Segmentation has basically two parts, such as line segmentation and character segmentation. Text line segmentation was done using horizontal projection profile. Sinhala handwritten character segmentation is difficult task, unlike printed characters; human written documents can be included more overlapping and touching characters. Figure 1.1 is shown overlapping and touching character groups.

Figure 1.1: Character groups (a) overlapping, (b) touching

Overlapping characters cannot be segmented using vertical line; each character has overlapped each other. Figure 1.2 is shown straight line in between overlapping characters.

This research was used the connected pixels labeling method to segment the overlapping characters [10]. After overlapping character segmentation, some characters were identified as isolated and some characters were identified as touched. Touching character segmentation was done using top peak and bottom valley points identification method [6], [11].

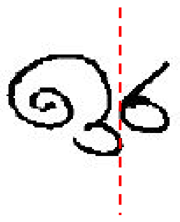


Figure 1.2: Straight line in between overlapping characters

Each segmented character was classified using preliminary classification. Character image was divided into eight horizontal and eight vertical strips. It produced 64 zones. Feature vector size is 65 dimensions (classification group with 64 zones). Finally, character recognition was done using linear support vector machine.

1.5 Chapter Summary

This chapter discussed background and motivation, aim and objectives and solution of this research.

Chapter 2

Review of Literature

2.1 Chapter Introduction

This section describes about the methods of previous researches regarding to the segmentation, classification, feature extraction and recognition of handwritten characters.

2.2 Segmentation using Projection Profile

In some of the researches [2], [5], [12], [13], the segmentation of character was done using projection profiles. That segmentation was divided into three sections. They are line, word and isolated character segmentations.

Two consecutive text lines (it is assumed that text lines are not overlapping or touching) were segmented using horizontal projection profile. The horizontal projection profile is defined as Equation (2.1).

$$h(i) = \sum_{i=0}^{n} p(i,j) \qquad (2.1)$$

Each word has been segmented using vertical projection profile (it is assumed that two words are not overlapping or touching). Again, vertical projection profile has used to segment characters of each word. If overlapping or touching characters are present, they are identified as one single character. The vertical projection profile is defined as Equation (2.2).

$$v(j) = \sum_{j=0}^{n} p(i,j) \qquad (2.2)$$

Where p(i,j) is the pixel value which is either 0(black color pixel) or 255(white color pixel) in a black and white image. i and j is called as row and column value of the pixel respectively [12].

Some recent research, overlapping and touching characters were not perfectly segmented using vertical projection profile. They have been distinguished using average character width process [5].

Some of the researches have been used the horizontal and vertical projection profile; these were not considered about overlapping and touching character segmentation [2], [7].

Horizontal projection profile only depends on upper, lower lines and upper, lower baselines. Vertical projection profile only depends on word boundaries and character boundaries [2].

Arabic character recognition research has discussed this method as one of a segmentation method [12].

The advantage of this method is that projection profiles are independent of shape, size, and font of the character which does not until overlapping or touching.

A disadvantage of this method is that it cannot segmentation of overlapping, touching characters. If some collected handwritten documents have included skew characters that cannot be segmented from this method. It is another disadvantage of this method [12].

2.3 Segmentation using Pixel Cluster Detection Technique

To improve the Hindu language handwritten character recognition accuracy was done some researches. Horizontal and vertical projection profile could not be segmented overlapping and touching characters of Hindu script. A research in India was done to segmentation of conjunct and overlapping characters in Devanagari script on Hindi language using pixel cluster detection technique.

Devanagari script has the header line of each word. First, that horizontal header line is removed using horizontal projection profile. Header line is identified using maximum number of black pixels. After detected the header line, that maximum number of black pixels are converted to white. Then the gap between each character of the word is calculated, and that value is stored into an array. After that mid value of each gap is calculated and that value is stored into another array. When two or more characters are touched or overlapped, the difference between adjacent mid values increase the decided number, it is taken as a cluster. Again, mid values of that heap of pixels (cluster) are calculated. Finally, characters are segmented using that mid values by drawing vertical line between two characters.

This proposed method is used to isolated, conjuncts, overlapping and touching handwritten characters' segmentation of Hindu script. It has got some higher accuracy to segmentation of characters [8].

An advantage of this technique is that it can be segmented the conjuncts, overlapping and touching handwritten characters than projection profiles.

Pixel cluster detection technique takes more time to segmentation of characters than horizontal and vertical projection profile.

2.4 Segmentation using Water Reservoir Based Approach

Water Reservoir based approach was used to segmentation of handwritten touched numerals and Sinhala characters in some researches [6], [11].

It was introduced the new concept to obtain features based on water reservoir. Reservoir is obtained place where to water flow from top or bottom of the component to identify the touching position such as top, middle or bottom [11], [14], [15].

At first, number of reservoirs, their size, position and location are identified to isolated and touching digit separately. Then reservoir boundary, touching position, topological features of touching pattern of identified touching digit are analyzed to determine the best cutting point. Finally the best cutting point is combined with morphological structural features and segmentation path is generated.[11].

Basic advantage of this approach is that it is segmentation of handwritten touched numerals and characters. Another advantage is that it is not depend on the size; the component size is not needed to normalization.

The drawback of the approach is that cutting point of the contour same as to boundary of the reservoir. Because of that segmentation is not properly done. Another drawback is that when double touching numerals or characters will be met, errors can be obtained.

This approach is rejecting in some points. They are: when the widths of two segmented numerals are different, when the length of the breaking path is larger than height of the touching pattern, and when the best reservoir of touching numeral is not obtained [11].

2.5 Segmentation using Self-Organizing Feature Maps

A journal paper has presented an intelligent technique to segmentation of touched characters in handwritten English words. From this study, the touching section of the cursive words was identified by Self-Organizing Feature Maps (SOM).

First in this work, scanned connected character document (image) is preprocessed. Then ascenders and descenders of touched components have found by core-zone estimation. As feature vector, each pixel of the image is mapped to coordinate system. Then feature vector is clustered into three regions. They are left, middle and right. Next middle region is used to vertical segmentation using SOM [16].

Benefits of SOM segmentation is that it can be segmented single and multiple touched characters better than the water reservoir approach. To improve the performance SOM segmentation, core-zone estimation can be applied. Moreover, when set the iterations within five iterations, cluster process computation time can be minimized.

Drawbacks of the technique is that SOM segmentation is only considered middle region based on the black pixels; middle region has large number of black pixels and small number of white pixels. Because of that vertical segmentation can be incorrect [16].

2.6 Classification using Statistical Classification Approach

Some of the research describes Sinhala handwritten character recognition from statistical approach. This research classification part was done using statistical classification approach, it is called preliminary classification. Its' aim is that number of possible candidates is reduced to an unknown character in the total character set. Each character is categorized into one of the six pre-classification groups. They are core character, core modifier, and ascending character, descending character, upper modifier and lower modifier. Only this groups of characters are considered for the recognition in that research [2], [17].

Advantage of this approach is that pre-classification is improved the accuracy of recognition the handwritten characters [2].

2.7 Feature Extraction Technique

Segmented characters of the image document are resized into 32x32 pixel size using bilinear interpolation technique.

One of the researches, each document is divided into horizontal and vertical strip lines. Then each strip line is again divided into 4x4 pixels. Using pixel density of each pixel, feature vector is created in horizontal and vertical directions [7].

Another research, a 71-dimentional feature vector is created for each resized image. In feature vector, first three features are pre-classification group, height and width. Next four features are included top, bottom, left and right of pixel densities of the image. When image is divided into 8x8 pixels, it produces equal sized 64 zones. Next features are included pixel densities of each equal sized 64 zones. Finally 4x4 pixel zones are used to get best results for feature extraction [2].

From above two feature extraction techniques, a 71-dimentional feature vector is better than the horizontal and vertical direction feature vector.

2.8 Character Recognition using Hidden Markov Models

Most of the researches has used this method to recognition of the handwritten characters. HMM is mostly used for speech recognition. Recently it is used for printed and handwritten character recognition [18]. They are Sinhala and Tamil handwritten character recognition researches [7], [9].

Different range in series of data is modeled from Hidden Markov Model that is statistical tool. HMMs have used to success of problems such as speech tagging and phrase chunking in natural language processing [19].

In Sinhala handwritten character recognition, two HMMs are created to horizontal data modelling and vertical data modelling. Standard procedures are used to train the Hidden Markov character Models [20].

Advantages of the HMM is that it is strong statistical approach and efficient learning algorithm. Another advantage is that it can handle most flexible variable length as inputs.

Disadvantages of HMM is that it has lot of unstructured parameters, first order HMMs are limited from their Markov property [21].

2.9 Character Recognition using Artificial Neural Network

Some of Sinhala character recognition researches were done based on the Artificial Neural Network (ANN). That method was used in the most of the sections of character recognition researches such as segmentation, feature extraction, classification [3], [22], [23].

In a research, after rescaling and feature extraction, this technique is used to recognize the characters [4].

Advantages of ANN is that it is easy to use than HMM. ANN is good method to recognize the handwritten characters.

Disadvantages of ANN is that it takes more time to train the system than HMM.

2.10 Chapter Summary

This chapter discussed others work in previous researches. Some of the researches are more prominent than others.

Chapter 3

Technology Adapted

3.1 Chapter Introduction

Segmentation of Overlapping Sinhala Handwritten Characters were done using image processing techniques. Mainly OpenCV 2.4.11 software was used to take the digital image processing techniques. In Microsoft Visual Studio 2012, C++ language were used to implement the algorithm.

3.2 OpenCV library

OpenCV (Open Source Computer Vision Library) is released under a BSD license. It is free to use both academic and commercial. C++, Python and Java interfaces are existed in OpenCV. It supports Windows, Linux, Mac OS, iOS and Android. OpenCV is strongly focused on real time applications. It is designed for computational works. OpenCV is written in optimized C/C++.The OpenCV library can take advantage of multi-core processing. It is enabled with OpenCL. OpenCV is adopted all around the world. More than 47 thousand people are used OpenCV. It is estimated number of downloads exceeding 14 million [24].

3.3 Development Tool

Microsoft Visual Studio is an Integrated Development Environment (IDE) from Microsoft. Computer programs, web sites, web apps, web services and mobile apps can be developed using MS Visual Studio. It uses Microsoft software development platforms. They are Windows API, Windows Forms, Windows Presentation Foundation, Windows Store and Microsoft Silverlight. Both native code and managed code can be produced it [25].

3.4 Programming Language

C++ is a general purpose programming language. Imperative, object-oriented and generic programming features can be existed in C++, while low-level memory manipulation is being provided the facilities [26].

3.5 Chapter Summary

This chapter discussed technologies that adapt to solve the research problem.

Chapter 4

Analysis and Design

4.1 Chapter Introduction

This section describes the details of analysis and design of my solution. Proposed system has six main sections. They are image acquisition, preprocessing, segmentation, classification, feature extraction and recognition. Top Level Diagram of this research is shown in Figure 4.1.

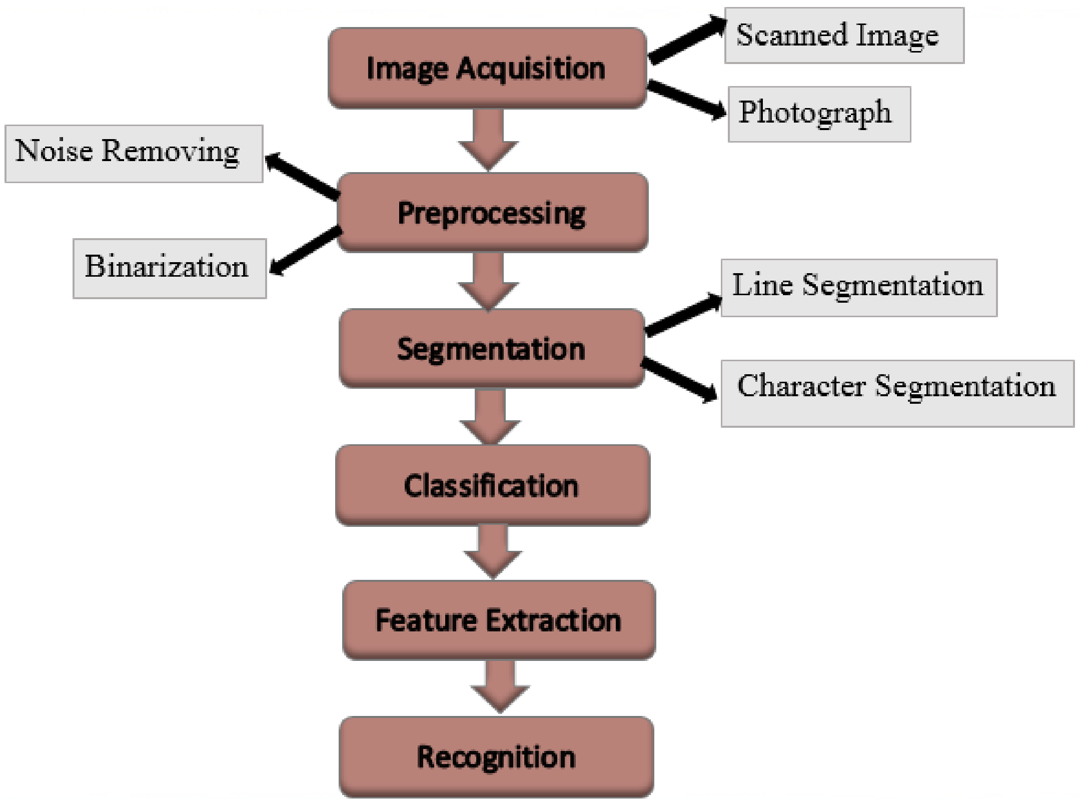


Figure 4.1 Top Level Diagram of the Proposed System

4.2 Image Acquisition

A4 sized paper is used to collect the sample handwriting. Sample document has 8 – 10 text lines and 3 – 4 words in each line. That document is scanned to get the image from JPEG, PNG or BMT format. This image can be taken from digital camera, scanner or any other digital input device.

4.3 Preprocessing

In the beginning, noise is removed the 8-bit gray scale image using median filter that replaces the pixel values from the median value of their neighborhoods. Then filtered gray image is converted to binary format (black and white image) using Binary thresholding method. White color pixels are foreground pixels and black color pixels are background pixels. It is shown in Figure 4.2. Finally, black pixel line (background pixel line) is created around image. Because some errors can be generated, when first, last row pixels and first, last column pixels are compared with top, left and bottom, right pixels.

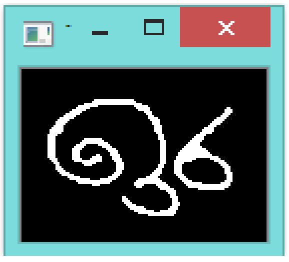


Figure 4.2 Binarization (thresholding)

4.4 Segmentation

In my approach, segmentation has two sections.

- Line Segmentation
- Character Segmentation

4.4.1 Line Segmentation

Text lines of preprocessed image were segmented using horizontal projection profile. Each text line was extracted among zero values or valley points of horizontal projection profile. Figure 4.3 is shown the horizontal projection profile of each line.

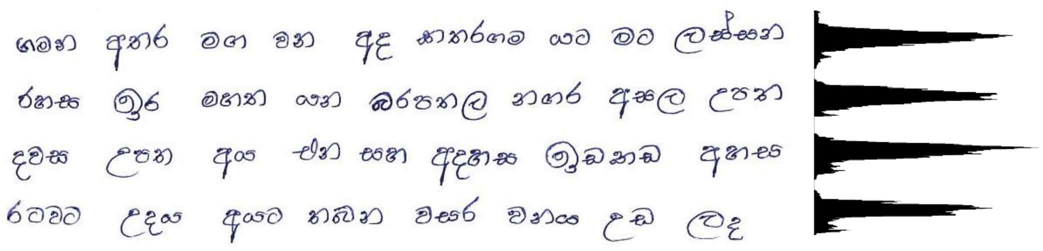
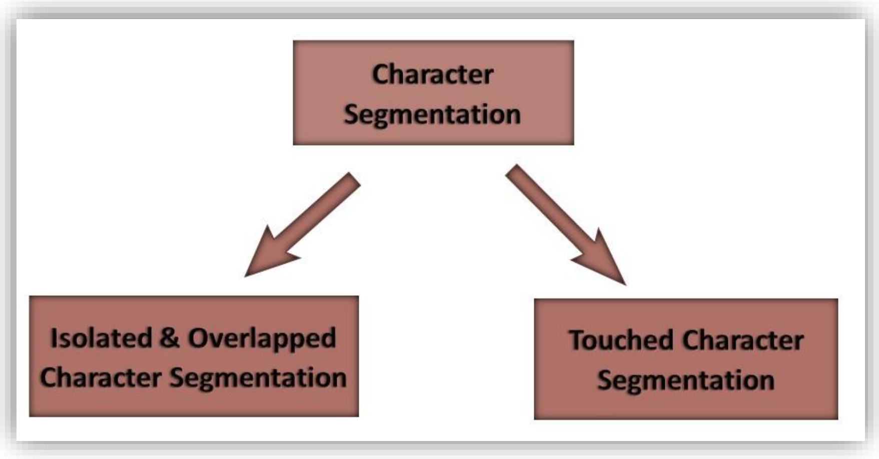


Figure 4.3: The horizontal projection profile of each line

4.4.2 Character Segmentation

After the line segmentation, each character is segmented using Character segmentation. It has 2 parts. It is shown in Figure 4.4

1. Isolated and overlapping character segmentation
2. Touching character segmentation



Character Segmentation
Isolated & Overlapped Character Segmentation
Touched Character Segmentation

Figure 4.4: Classification of Character Segmentation

4.4.2.1 Isolated and Overlapping Character Segmentation

Overlapping character segmentation was done using connected pixel labelling method. This method included in “Segmentation of Overlapping and Touching Sinhala Handwritten Characters” [27] IEEE research paper in 2018 which was written by me.

Pixels of part of the binary image are taken to describe my isolated and overlapping character segmentation algorithm. Figure 4.5 is shown a block that represent the pixels of part of the binary image.

First, a matrix is created to store the black and white pixels of whole image. Figure 4.6 is represented the matrix that shows the black pixels as ‘0’ (background) and the white pixels as ‘1’ (foreground).

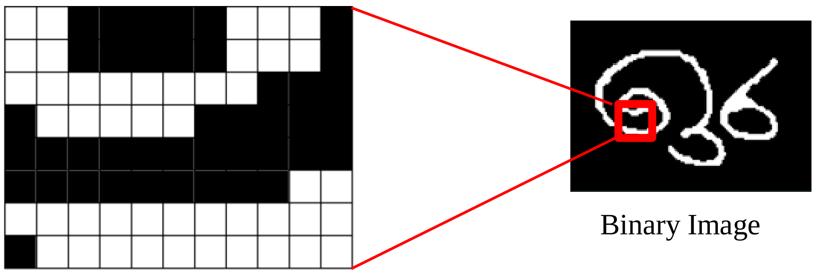


Figure 4.5: A block that represents the pixels of part of the binary image

1	1	0	0	0	0	0	1	1	1	0
1	1	0	0	0	0	0	1	1	1	0
1	1	1	1	1	1	1	1	0	0	0
0	1	1	1	1	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	1	1
1	1	1	1	1	1	1	1	1	1	1
0	1	1	1	1	1	1	1	1	1	1

Figure 4.6: A matrix for the pixels of part of the binary image

There are four rounds of the isolated and overlapping character segmentation. They are first round, second round, third round and fourth round.

First Round

Pixel by pixel is read using two for loops. If pixel is background, it is assigned same black color to image and number '0' to matrix. But if pixel is foreground ('1'), it is considered left and top pixel.

Step 1:

If a pixel has not left or top pixel or both left or top pixel is background, Then, new number and color is applied to matrix for corresponding pixel.

eg:- first pixel (matrix [0][0]) is shown in Figure 4.7. It is foreground pixel. Both left and top pixels are not existing. New number with new color is applied to first pixel.

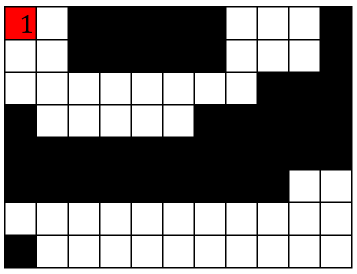


Figure 4.7: The matrix that represents Step 1 of First Round

Step 2:

If left pixel exists and top pixel doesn't exist, number and color of left pixel is applied to second pixel.

eg:- Second pixel (matrix [0][1]) is shown in Figure 4.8

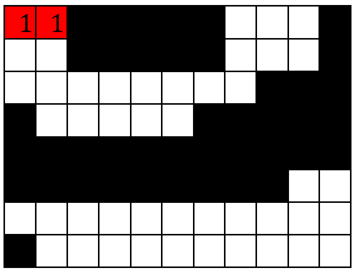


Figure 4.8: The matrix that represents Step 2 of First Round

3rd, 4th, 5th, 6th, 7th pixels are background pixels. 8th pixel (matrix [0][8]) is foreground pixel. Its left pixel is background pixel and top pixel doesn't exist. Then new number and color can be applied to that pixel. 9th and 10th pixels are foreground pixels. Both pixels have left pixels and haven't top pixels. left pixel number '2' (color green) can be applied to that pixels. 11th pixel is background pixel. These things are shown in Figure 4.9

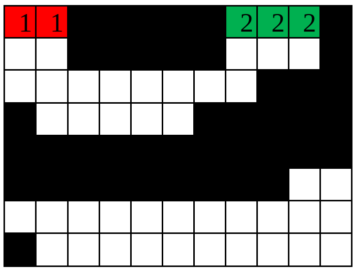


Figure 4.9: The matrix with Some Changes

Step 3:

If left pixel doesn't exist, top pixel exist, number and color of top pixel is applied to the pixel.

eg:- Second row first pixel (matrix [1][0]) is shown in Figure 4.10. Second row and third row, first 7 pixels are being filled as Figure 4.11

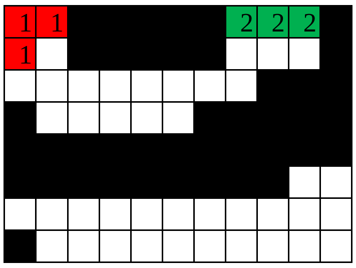


Figure 4.10: The matrix that represents Step 3 of First Round

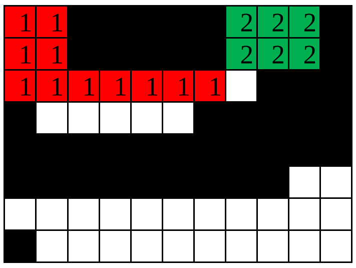


Figure 4.11: The matrix with Some Changes

Step 4:

If both left and top pixels have different foreground pixel numbers with colors, smaller number should be applied with its color to corresponding pixel. Also, small number as parent and large number as child should be stored in a data structure.

eg:- Third row, 8th pixel (matrix [2][7]) has both left and top pixels with different numbers and colors. Smaller number with color is applied to that pixel. Also, small number ('1') is stored as parent and large number ('2') is stored as child in a pair array. That is shown in Figure 4.12.

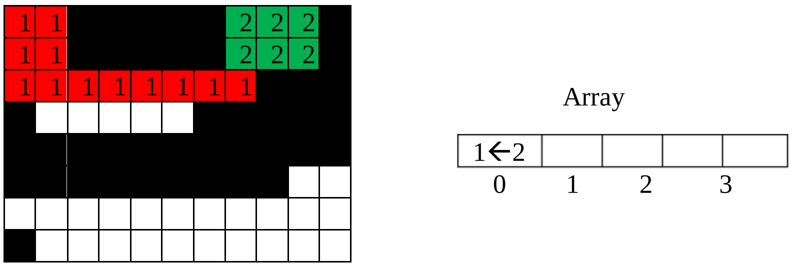


Figure 4.12: The matrix that represents Step 4 of First Round

As figure 4.13, 4th row can be filled. Whole 5th row is background. 6th row, first 9 pixels are background. 10th pixel (matrix [5][9]) is applied new number with new color (because both left and top are background). Last pixel (10th pixel) of 6th row is applied same number with color. These things are shown in Figure 4.13.

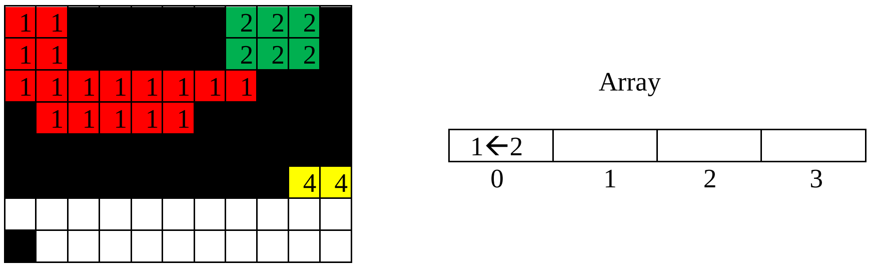


Figure 4.13: The matrix with Some Changes

7th row, 1st pixel is applied new number with new color. Then next 8 pixels are being filled like following image. Next pixel (matrix [6][9]) has different left and top pixels. Smaller number with color can be applied for that pixel. parent and child number are stored to the array. Final result of the matrix is shown in Figure 4.14. Result of after completed the first round for all the pixels of the image is shown in Figure 4.15

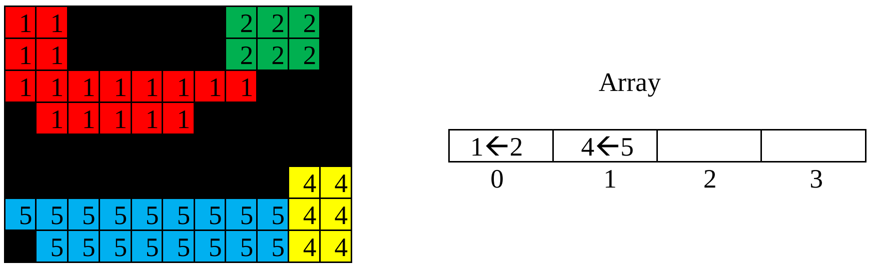


Figure 4.14: The matrix with Final Result of First Round

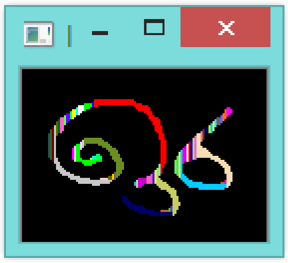


Figure 4.15: Result of after completed the first round for all the pixels of the image

Second Round - Step 1

Again, pixel by pixel is read using loops. Each pixel is checked whether background pixel or not. If pixel is not a background pixel, then pixel is compared with child numbers of the pair array. If pixel is equal to a child number, then its parent number with color is applied to corresponding pixel.

When pixel by pixel is read of first row, '1' is not child. But '2' is child of '1' (parent). Then number '2' with green color is replaced from '1' with red. That is shown in Figure 4.16

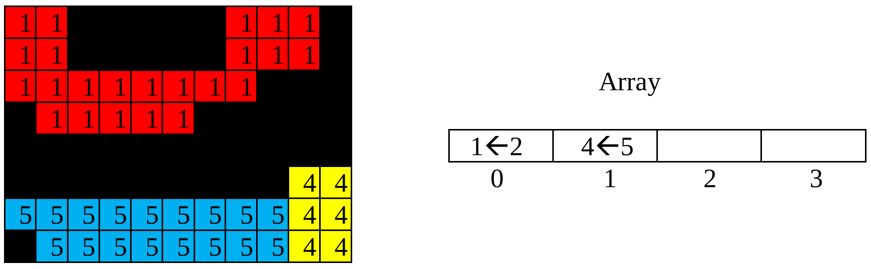


Figure 4.16: The matrix that represents the replaced number '1' with red

Number '4' is not child. But number '5' is child of '4'. So, number '5' with blue color is replaced from '4' with yellow. Figure 4.17 is shown the part of the pixel set is divided into two connected sections. But when it is applied to character image, whole character is not segmented completely. It is shown in Figure 4.18

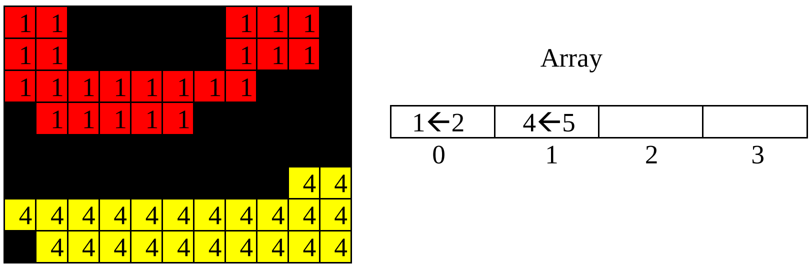


Figure 4.17: The matrix that represents part of the pixel set of second round step 1 completed image

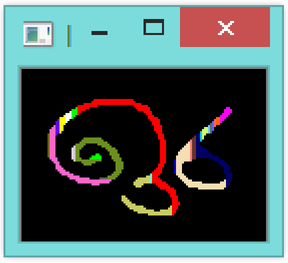
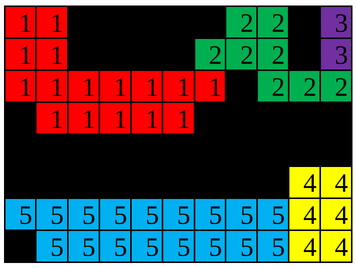


Figure 4.18: Result of after completed the step 1 of second round

for all the pixels of the image

Figure 4.19 is shown another some set of pixels are taken from the completed image of first round. Figure 4.20 is shown that set of pixels are completed the step 1 of second round. That connected pixels are not segmented completely. The solution for that problem is described in Step 2 of Second Round.

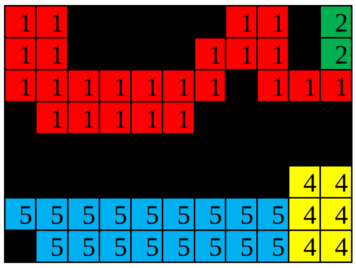


1	1						2	2		3
1	1					2	2	2		3
1	1	1	1	1	1	1		2	2	2
	1	1	1	1	1					
									4	4
5	5	5	5	5	5	5	5	5	4	4
	5	5	5	5	5	5	5	5	4	4

Array

1←2	2←3	4←5	
0	1	2	3

Figure 4.19: Some set of pixels are taken from the completed image of first round



1	1						1	1		2
1	1					1	1	1		2
1	1	1	1	1	1	1		1	1	1
	1	1	1	1	1					
									4	4
5	5	5	5	5	5	5	5	5	4	4
	5	5	5	5	5	5	5	5	4	4

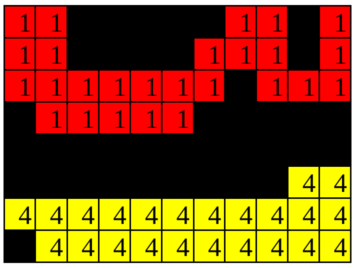
Array

1←2	2←3	4←5	
0	1	2	3

Figure 4.20: Some set of pixels are completed the step 1 of second round

Second Round - Step 2

According to figure 4.20, number ‘1’ is parent of number ‘2’. Number ‘2’ is parent of number ‘3’. To rectify this error, second-round algorithm should be applied again. Output is shown in Figure 4.21. Like this, second-round algorithm can be applied until not any change. It is shown in Figure 4.22.



Array

1←2	2←3	4←5	
0	1	2	3

Figure 4.21: Some set of pixels are completed the step 2 of second round

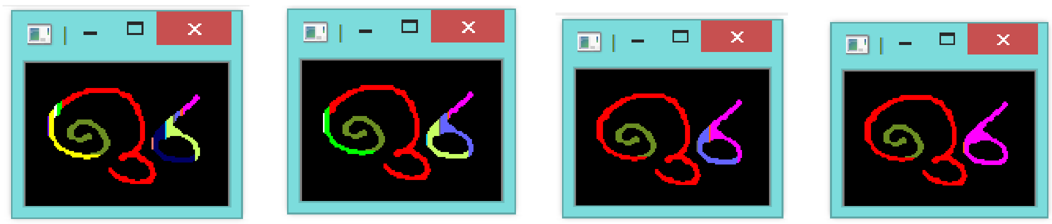
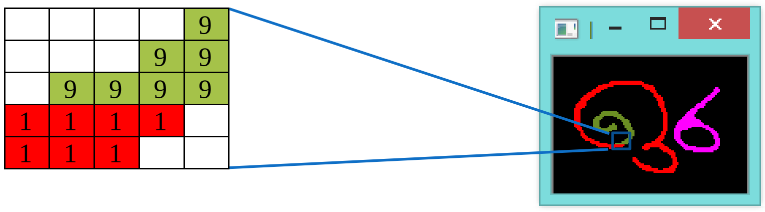


Figure 4.22: The output images those are applied Second Round Algorithm until not any change

Third Round

After second round is finished, some characters are not segmented completely. Thus, a character has two colors or two numbers. When the array is checked, both two numbers are parents. Some example is shown in Figure 4.23



Array

1←2	2←3	4←5	9←10
0	1	2	3

Figure 4.23: Some set of pixels are taken from a non-segmented character, after completed second round

To rectify this error, pixel by pixel is read again and compared with right pixel and bottom pixel.

If a pixel is smaller than right pixel, then the pixel number with color is stored as parent and right pixel number with color is stored as child in a new pair array.

Otherwise, right pixel number with color is stored as parent and pixel number with color is stored as child in the new pair array.

If a pixel is smaller than bottom pixel, then the pixel number with color is stored as parent and bottom pixel number with color is stored as child in the new pair array.

Otherwise, bottom pixel number with color is stored as parent and pixel number with color is stored as child in the new pair array. Figure 4.24 is shown the new array to store the parent and child number of set of pixels of example Figure 4.23

New Array

1←9			
0	1	2	3

Figure 4.24: A new Array that represents after completed the third round

Fourth Round

Each pixel is read one by one. If a pixel is foreground, then it is compared with child values of the new array.

If a pixel is equal to a child number with color, then its parent number with color is applied for that pixel. Figure 4.25 is shown the set of pixels, after completed fourth round. Figure 4.26 is shown final output of completely segmented characters of entire image.

				1
			1	1
	1	1	1	1
1	1	1	1	
1	1	1		

Figure 4.25: A set of pixels, after completed the fourth round

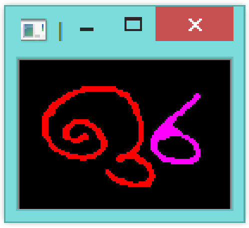


Figure 4.26: final output of segmented characters of entire image

4.4.2.2 Touching Character Segmentation

In overlapping character segmentation, touching characters were taken as one character. An example is shown in Figure 4.27.

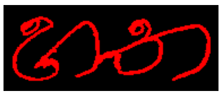


Figure 4.27: A touching character image after overlapping character segmentation

Touching character segmentation was done using peak and valley point identification method. This method included in "Segmentation of Overlapping and Touching Sinhala Handwritten Characters" [27] IEEE research paper in 2018 which was written by me.

overlapping segmented character was checked whether it was isolated or touching. To detect it, peak or valley points were identified. Before the peak or valley point identification was applied, character was cropped within boundaries. Then, first-black pixel of each column was taken and drawn a histogram as Figure 4.28. Peak points were used to find top touching point. Last-black pixel of each column was taken and drawn a histogram as Figure 4.29. Valley points were used to find bottom touching point.

Minimum to maximum column number that touched valleys between a peak was identified as a top range as shown in Figure 4.28. Minimum to maximum column number that touched lower peaks between a valley was identified as a bottom range as shown in Figure 4.29. First and last peak of Figure 4.28 were not considered as top ranges. First and last valley of Figure 4.29 were not considered as bottom ranges. According to histograms, three top ranges and three bottom ranges were identified as Figure 4.30.

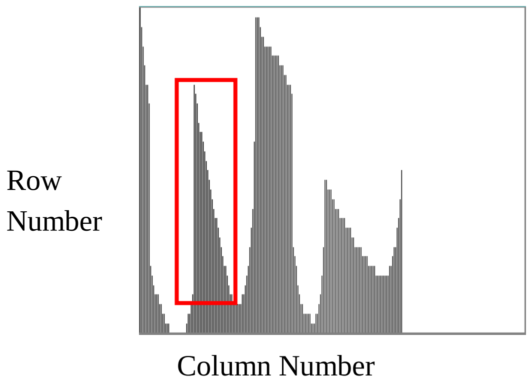


Figure 4.28: Histogram of the first black pixel of each column

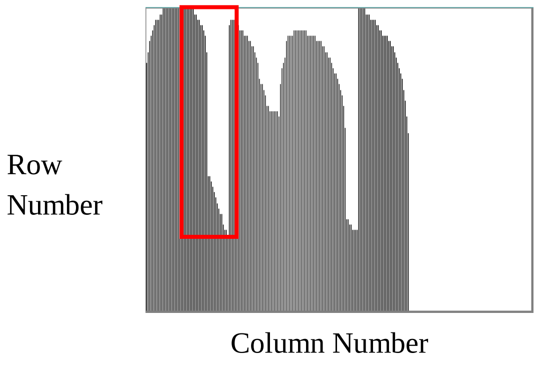


Figure 4.29: Histogram of the last black pixel of each column

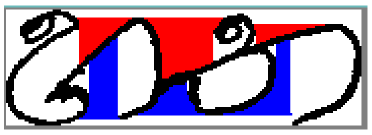


Figure 4.30: Three top and three bottom ranges

Touching character segmentation has three steps. In first step, matching ranges were found in between top ranges and bottom ranges. If number of matching ranges were greater than zero, then second step was applied. Otherwise character was isolated. It is shown in Figure 4.31(a).

In second step, ranges were found that without white pixels in between the top and bottom ranges. If number of ranges without consist white pixels were greater than zero, then third step was applied. Otherwise character was isolated. It is shown in Figure 4.31(b).

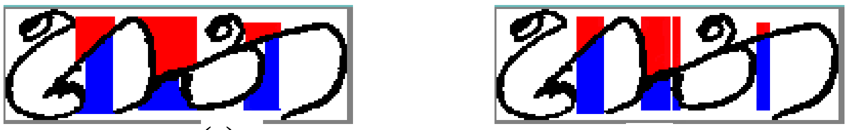


Figure 4.31: Touching character segmentation (a) first step, (b) second step

In third step, each range was checked for two conditions. They are; if black line was going down at the left to right of top range and going up at the left to right of bottom range, then a touching point was detected. Otherwise, if black line was going down at the right to left of top range and going up at the right to left of bottom range, then a touching point was detected. Otherwise this step was applied to next range. Figure 4.32 is shown the different points that was detected as touching point. Example of third step is shown in Figure 4.33. First, third ranges are not touching, and second range is touching. Number of detected touching points was greater than zero, then character was touching, otherwise character was isolated.

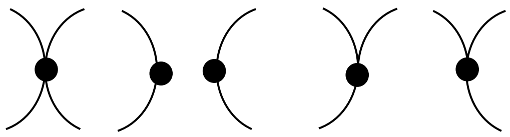
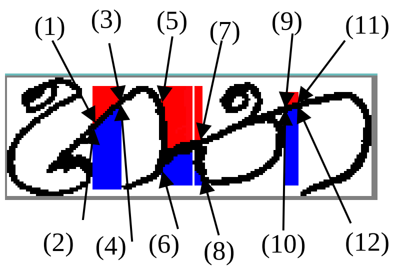


Figure 4.32: Points that detected as touching point



First Range	Second Range	Third Range
→ (1) Going up	→ (5) Going down	→ (9) Going up
← (2) Going up	← (6) Going up	← (10) Going up

Figure 4.33: Third step of Touching character segmentation

4.5 Classification

Segmented characters were classified using preliminary classification. After each text line segmentation, two highest pixel densities through the peaks of horizontal pixel densities were calculated and kept as baseline (upper and lower) locations. An example of upper and lower baselines of a segmented text line is shown in Figure 4.34. Upper and lower boundary lines of a character are shown in Figure 4.35. According to baselines and character boundary lines, there are three classification groups. They are ascending characters, core characters and descending characters as shown in Figure 4.36. Table 4.1 is shown the classified characters to each group.

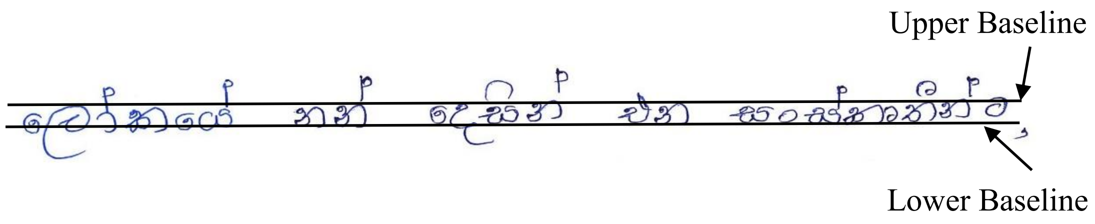


Figure 4.34: Upper and Lower Baselines of a Segmented Text Line

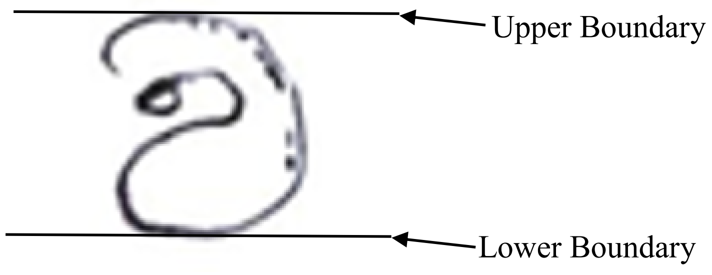


Figure 4.35: Upper and lower boundary lines of a character

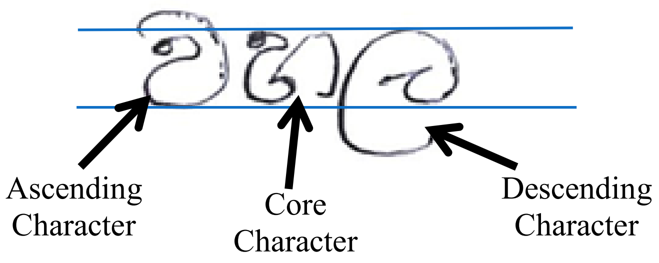


Figure 4.36: Three Classification Groups

Table 4.1: Classified Characters to Each Group

Group	Characters of each group	Number of characters
Core Characters	ප ස ය ක න ත ග හ	8
Ascending Characters	ර ඵ ඩ ව ට ඹ ම	7
Descending Characters	ඇ ඉ උ ද ල ළ	6

4.6 Feature Extraction

Each segmented character was rescaled into common width and height image (32 x 32 pixels). That image was divided into 8 horizontal and 8 vertical strips. It produced 64 zones. Size of each zone is 4 x 4 pixels. Example of 8 horizontal and 8 vertical strips for a segmented character is shown in Figure 4.37. A feature vector of each character was created using character classification group and pixel densities of 64 zones. This feature vector has 65-dimentions. It is shown in Figure 4.38.

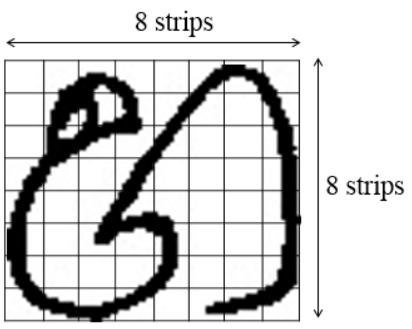


Figure 4.37: 8 horizontal and 8 vertical scrips for segmented character

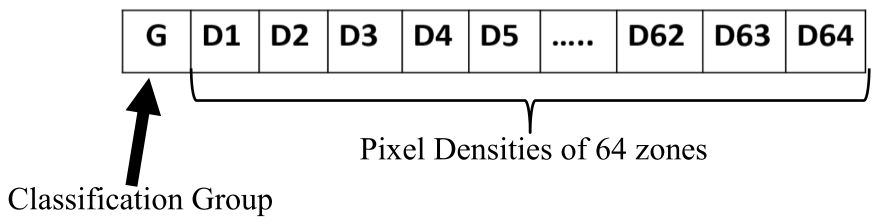


Figure 4.38: 65-dimentional Feature Vector

4.7 Recognition

Character recognition was done using a machine learning technique. It is Support Vector Machine (SVM). SVM is supported to classification and regression analysis [28]. Non-linear classification is performed efficiently. SVM build a hyperplane or set of hyperplanes in a high dimensional space or infinite dimensional space. It is used for classification, regression or outlier detection.

4.8 Chapter Summary

This chapter discussed preprocessing, line segmentation, new overlapping and touching character segmentation methods. Finally, it discussed how to do character classification, feature extraction and character recognition.

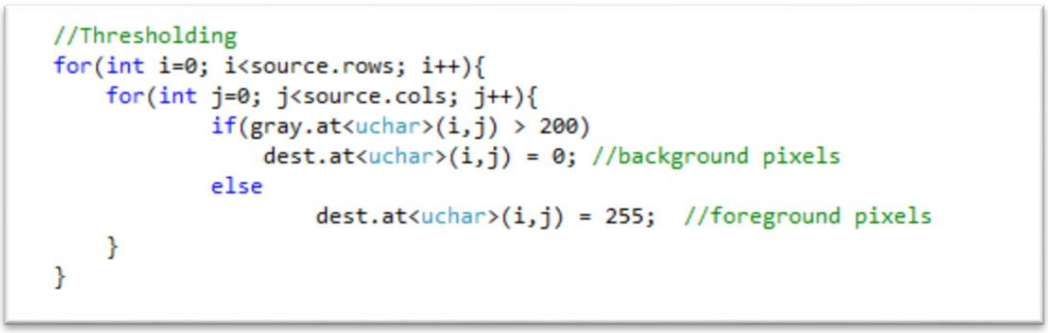
Chapter 5
Implementation

5.1 Chapter Introduction

This chapter describes source codes and algorithms for analysis and design in previous chapter.

5.2 Preprocessing

First scanned image is loaded to the system and converted to gray scale. Then gray scale image is filtered using median filter. Thresholding (Binarization) is done using convert filtered gray-scale image to black and white image. White color pixels (255) are foreground pixels and black color pixels (0) are background pixels. Figure 5.1 is shown that thresholding code segment.



```
//Thresholding
for(int i=0; i<source.rows; i++){
    for(int j=0; j<source.cols; j++){
            if(gray.at<uchar>(i,j) > 200)
                dest.at<uchar>(i,j) = 0; //background pixels
            else
                    dest.at<uchar>(i,j) = 255;  //foreground pixels
    }
}
```

Figure 5.1: Code segment of thresholding

After thresholding, Figure 5.2 is shown code segment that represents the creating black (0) pixel line the around image. Because some errors can be generated, when first, last row pixels and first, last column pixels are compared with top, left and bottom, right pixels.

5.3 Segmentation

Segmentation has two sections. They are line segmentation and character segmentation.

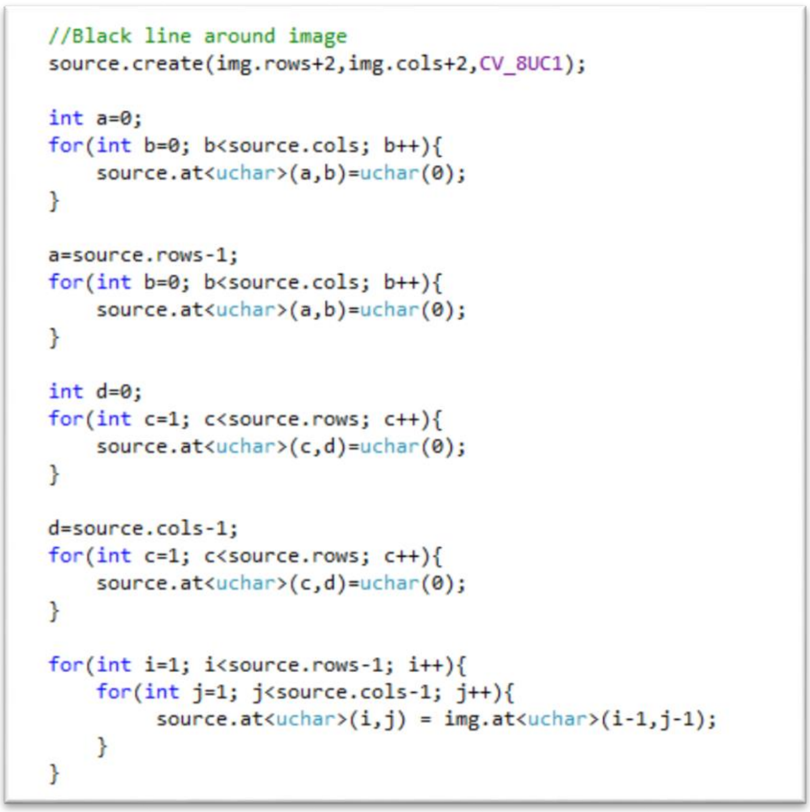
5.3.1 Line Segmentation

Horizontal projection profile of the binary image is generated from code segment of Figure 5.3. Segmented line of given line number is generated from code segment of Figure 5.4.

5.3.2 Character Segmentation

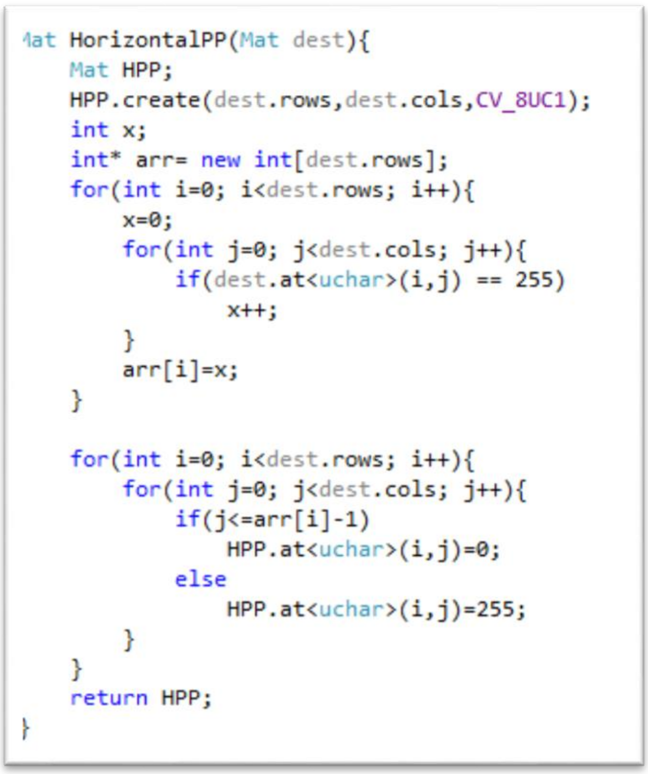
Character segmentation has two parts. Such as;

1. Isolated and overlapping character segmentation
2. Touching character segmentation



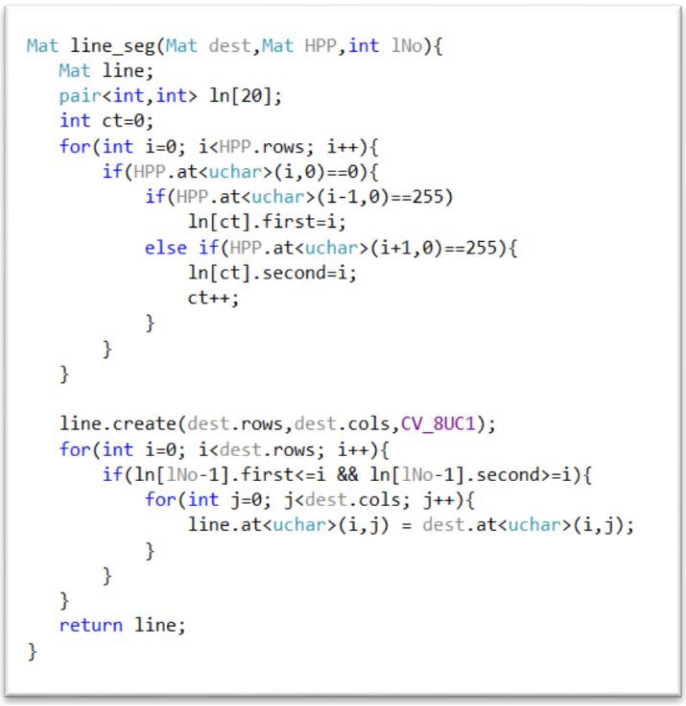
```
//Black line around image
source.create(img.rows+2,img.cols+2,CV_8UC1);

int a=0;
for(int b=0; b<source.cols; b++){
    source.at<uchar>(a,b)=uchar(0);
}

a=source.rows-1;
for(int b=0; b<source.cols; b++){
    source.at<uchar>(a,b)=uchar(0);
}

int d=0;
for(int c=1; c<source.rows; c++){
    source.at<uchar>(c,d)=uchar(0);
}

d=source.cols-1;
for(int c=1; c<source.rows; c++){
    source.at<uchar>(c,d)=uchar(0);
}

for(int i=1; i<source.rows-1; i++){
    for(int j=1; j<source.cols-1; j++){
        source.at<uchar>(i,j) = img.at<uchar>(i-1,j-1);
    }
}
```

Figure 5.2: Code segment of creating a black pixel line around image



```
Mat HorizontalPP(Mat dest){
    Mat HPP;
    HPP.create(dest.rows,dest.cols,CV_8UC1);
    int x;
    int* arr= new int[dest.rows];
    for(int i=0; i<dest.rows; i++){
        x=0;
        for(int j=0; j<dest.cols; j++){
            if(dest.at<uchar>(i,j) == 255)
                x++;
        }
        arr[i]=x;
    }

    for(int i=0; i<dest.rows; i++){
        for(int j=0; j<dest.cols; j++){
            if(j<=arr[i]-1)
                HPP.at<uchar>(i,j)=0;
            else
                HPP.at<uchar>(i,j)=255;
        }
    }
    return HPP;
}
```

Figure 5.3: Code segment of Horizontal projection profile of binary image

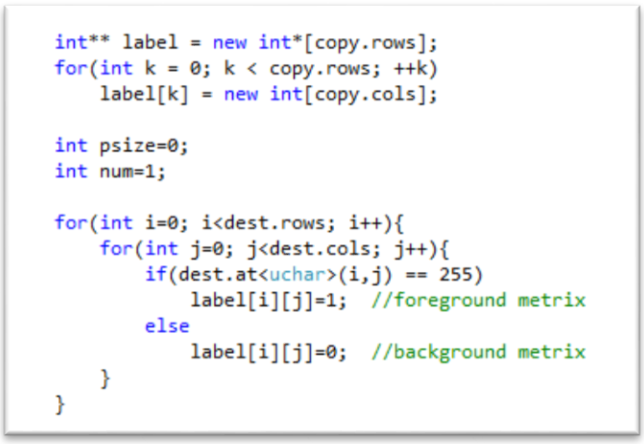


```
Mat line_seg(Mat dest,Mat HPP,int lNo){
   Mat line;
   pair<int,int> ln[20];
   int ct=0;
   for(int i=0; i<HPP.rows; i++){
       if(HPP.at<uchar>(i,0)==0){
           if(HPP.at<uchar>(i-1,0)==255)
               ln[ct].first=i;
           else if(HPP.at<uchar>(i+1,0)==255){
               ln[ct].second=i;
               ct++;
           }
       }
   }

   line.create(dest.rows,dest.cols,CV_8UC1);
   for(int i=0; i<dest.rows; i++){
       if(ln[lNo-1].first<=i && ln[lNo-1].second>=i){
           for(int j=0; j<dest.cols; j++){
               line.at<uchar>(i,j) = dest.at<uchar>(i,j);
           }
       }
   }
   return line;
}
```

Figure 5.4: Code segment of Segmented line of given line

5.3.2.1 Isolated and Overlapping Character Segmentation

According to my algorithm, first, a matrix is created to store the black and white pixels of whole image. Figure 5.5 is shown the code segment to create the matrix that shows the black pixels as ‘0’ (background) and the white pixels as ‘1’ (foreground).



```
int** label = new int*[copy.rows];
for(int k = 0; k < copy.rows; ++k)
    label[k] = new int[copy.cols];

int psize=0;
int num=1;

for(int i=0; i<dest.rows; i++){
    for(int j=0; j<dest.cols; j++){
        if(dest.at<uchar>(i,j) == 255)
            label[i][j]=1;  //foreground metrix
        else
            label[i][j]=0;  //background metrix
    }
}
```

Figure 5.5:Code segment to create a matrix

There are four rounds of my algorithm. They are first round, second round, third round and fourth round.

Figure 5.6 is shown the procedure of First Round in a flow chart. Figure 5.7 is shown the procedure of Second Round. Figure 5.8 is shown the procedure of Third Round.

5.3.2.2 Touching Character Segmentation

Each segmented overlapping character was checked to find the touching point by peak and valley identification method. If total number of touching points is greater than zero, character is touching character. Otherwise character is isolated. Procedure of touching character segmentation is shown in Figure 5.9.

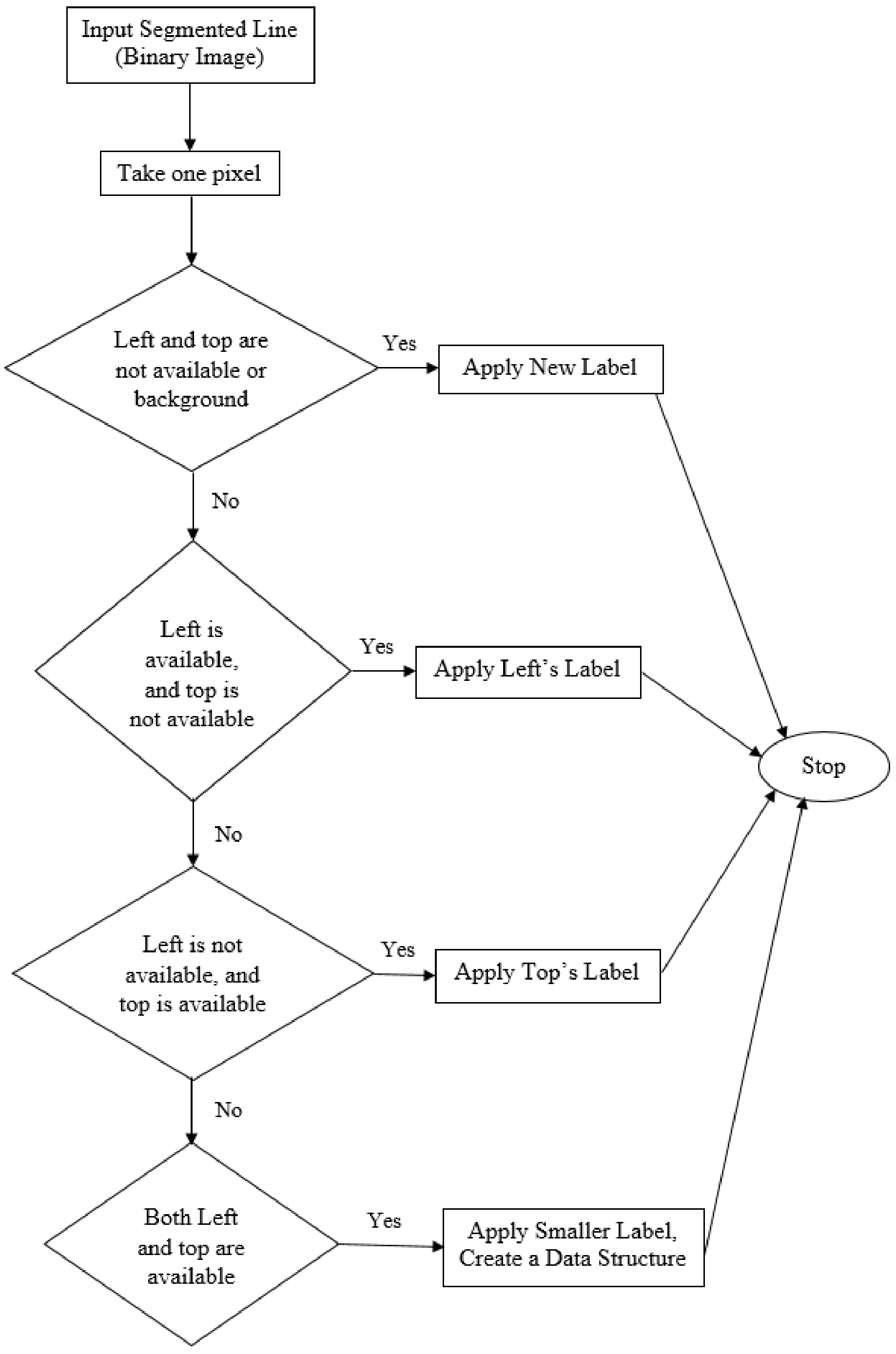

Input Segmented Line (Binary Image)
Take one pixel
Left and top are not available or background
Yes
Apply New Label
No
Left is available, and top is not available
Yes
Apply Left's Label
Stop
No
Left is not available, and top is available
Yes
Apply Top's Label
No
Both Left and top are available
Yes
Apply Smaller Label, Create a Data Structure

Figure 5.6: Procedure of First Round

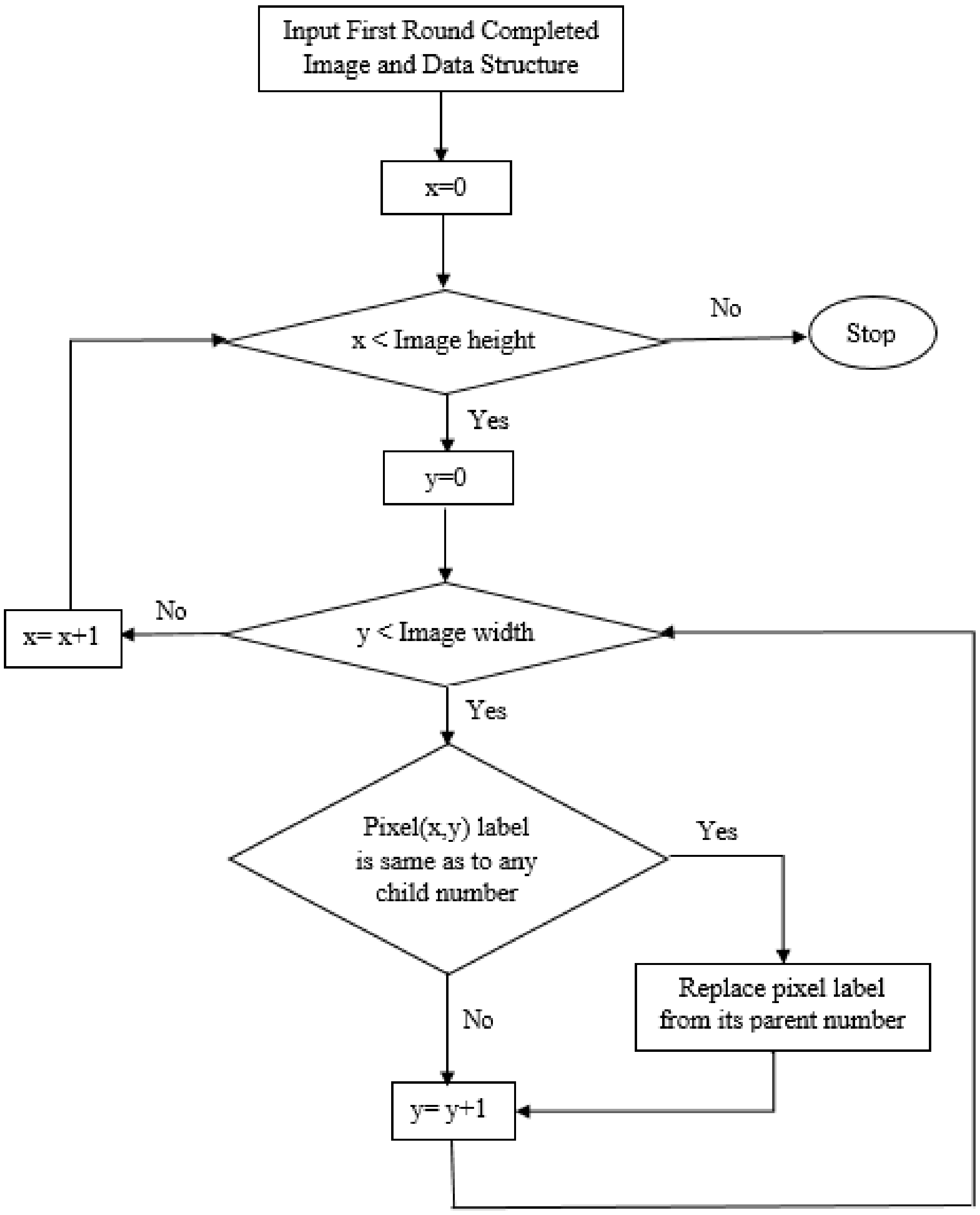

Input First Round Completed Image and Data Structure
x=0
x < Image height
No
Stop
Yes
y=0
y < Image width
No
x= x+1
Yes
Pixel(x,y) label is same as to any child number
Yes
Replace pixel label from its parent number
No
y= y+1

Figure 5.7: Procedure of Second Round

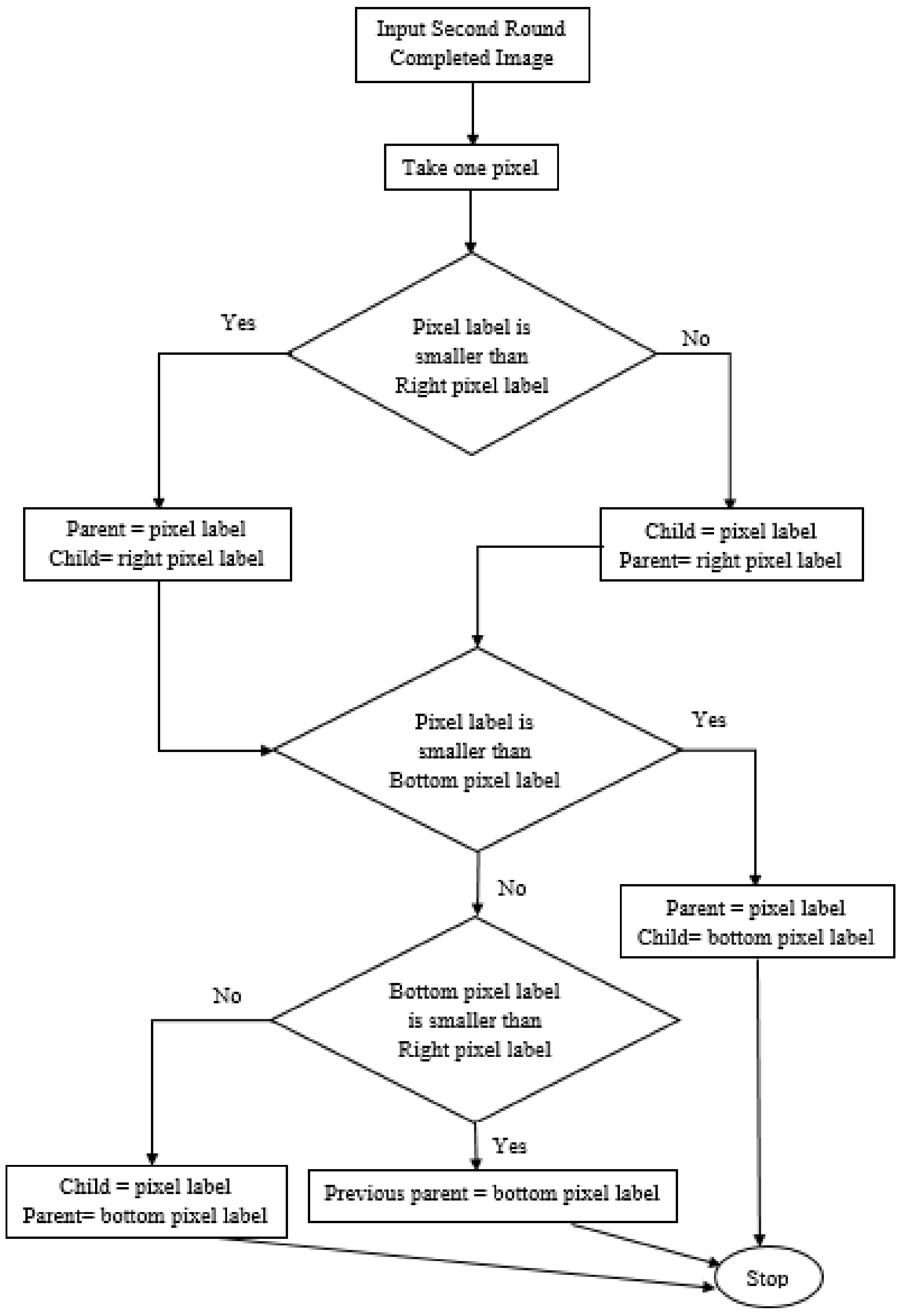

Input Second Round Completed Image
Take one pixel
Pixel label is smaller than Right pixel label
Yes
No
Parent = pixel label Child= right pixel label
Child = pixel label Parent= right pixel label
Pixel label is smaller than Bottom pixel label
Yes
No
Parent = pixel label Child= bottom pixel label
Bottom pixel label is smaller than Right pixel label
No
Yes
Child = pixel label Parent= bottom pixel label
Previous parent = bottom pixel label
Stop

Figure 5.8: Procedure of Third Round

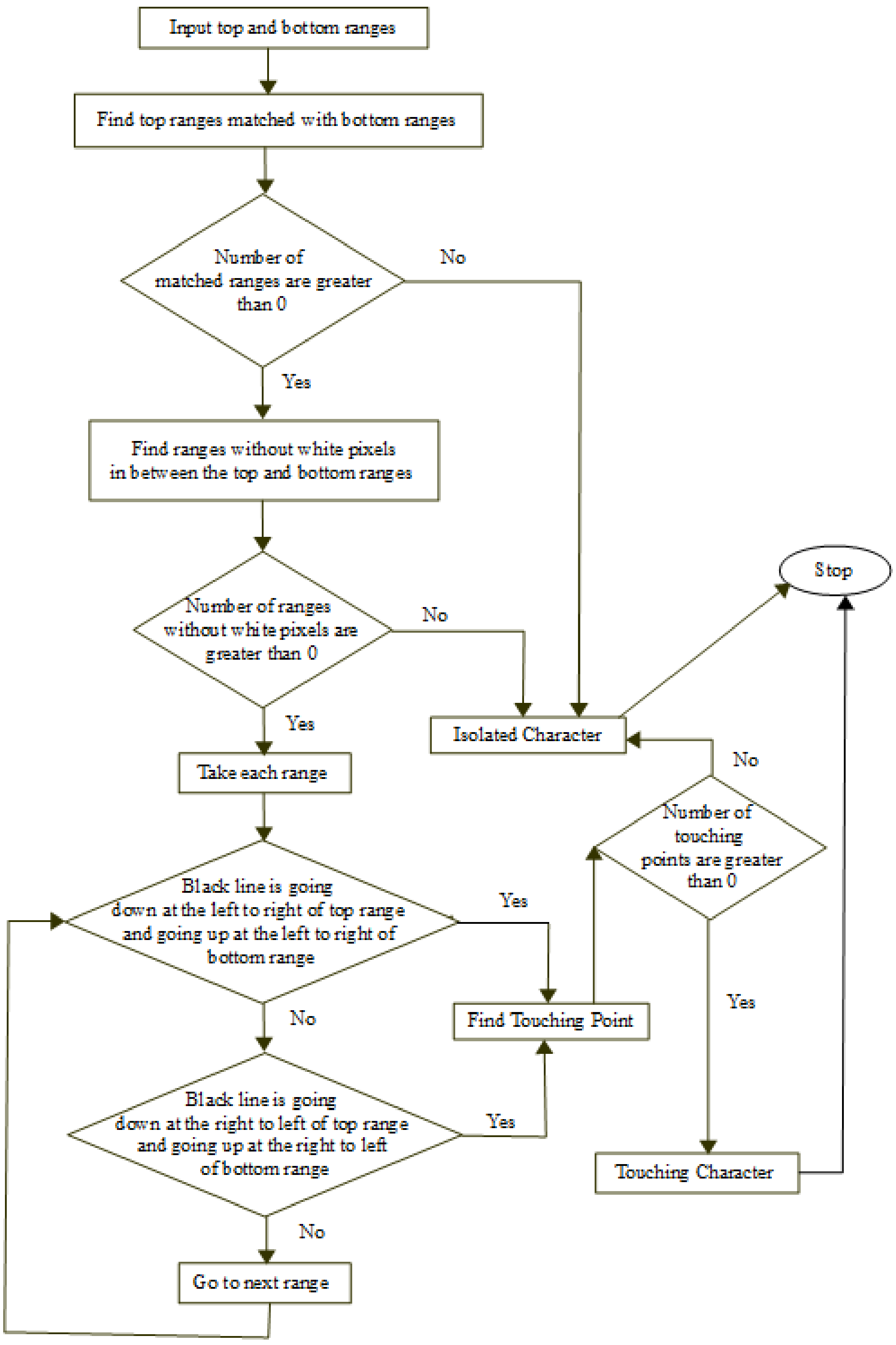

Input top and bottom ranges
Find top ranges matched with bottom ranges
Number of matched ranges are greater than 0
No
Yes
Find ranges without white pixels in between the top and bottom ranges
Number of ranges without white pixels are greater than 0
No
Yes
Stop
Isolated Character
Take each range
No
Number of touching points are greater than 0
Black line is going down at the left to right of top range and going up at the left to right of bottom range
Yes
Yes
Find Touching Point
No
Black line is going down at the right to left of top range and going up at the right to left of bottom range
Yes
Touching Character
No
Go to next range

Figure 5.9: Procedure of Touching Character Segmentation

5.4 Classification

After text line segmentation, peaks of horizontal pixel densities were found as the algorithm of Figure 5.10. Two highest pixel densities through the peaks were calculated to find the upper baseline and lower baseline as the algorithm of Figure 5.11. Upper baseline is upper row. Lower baseline is lower row. According to the baselines and character upper and lower boundary lines, preliminary classification was done as the algorithm of Figure 5.12.

```
R /*number of rows in text line*/
Ln[] /*pixel density of each line*/
cnt /*count*/
Pr[] /*row number of each peak*/
Pd[] /*pixel density of each peak*/

cnt = 0
x = 0
for x in R
    if Ln[x]>=Ln[x-1] and Ln[x]>=Ln[x+1]
        Pr[cnt] = x
        Pd[cnt] = Ln[x]
        cnt++
    end if
end for
```

Figure 5.10: Algorithm of find the peaks of horizontal pixel densities

5.5 Feature Extraction

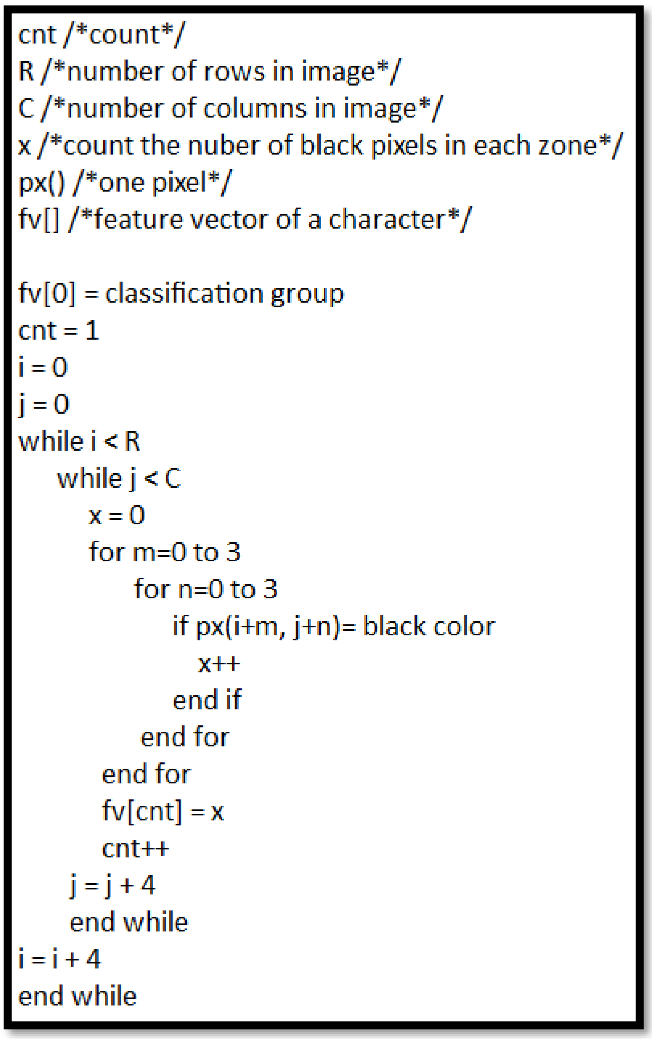
Each character image was rescaled into common width and height using OpenCV function. Then image was divided into 8 horizontal and 8 vertical strips and created 64 zones as the algorithm of Figure 5.13. Feature vector has a character classification group and 64 zones. Classification groups are ascending, core and descending characters. Ascending characters, core characters and descending characters represent as “1”, “2” and “3” respectively in feature vectors.

```
Max /*highest pixel density*/
cnt /*count*/
Pd[] /*pixel density of each peak*/
Pr[] /*row number of each peak*/
pk /*number of peaks*/
M[] /*row numbers of higher pixel densities*/
UB /*Upper Baseline of the text line*/
LB /*Lower Baseline of the text line*/

Max = 0
cnt = 0
y = 0
for y in pk
    if Pd[y]>Max
        Max=Pd[y]
        M[cnt]=Pr[y]
        cnt++
    end if
end for

if M[cnt-1]<M[cnt-2]
    UB = M[cnt-1]
    LB = M[cnt-2]
else
    UB = M[cnt-2]
    LB = M[cnt-1]
end if
```

Figure 5.11: Algorithm of find the upper baseline and lower baseline

```
UB /*Upper Baseline of the text line*/
LB /*Lower Baseline of the text line*/
upper /*upper boundary line of the character*/
lower /*lower boundary line of the character*/

if  upper=UB and lower=LB
      /* Core Character */
else if upper<UB and lower=LB
      /* Ascending Character */
else if upper=UB and lower>LB
      /* Descending Character */
end if
```

Figure 5.12: Preliminary Classification Algorithm

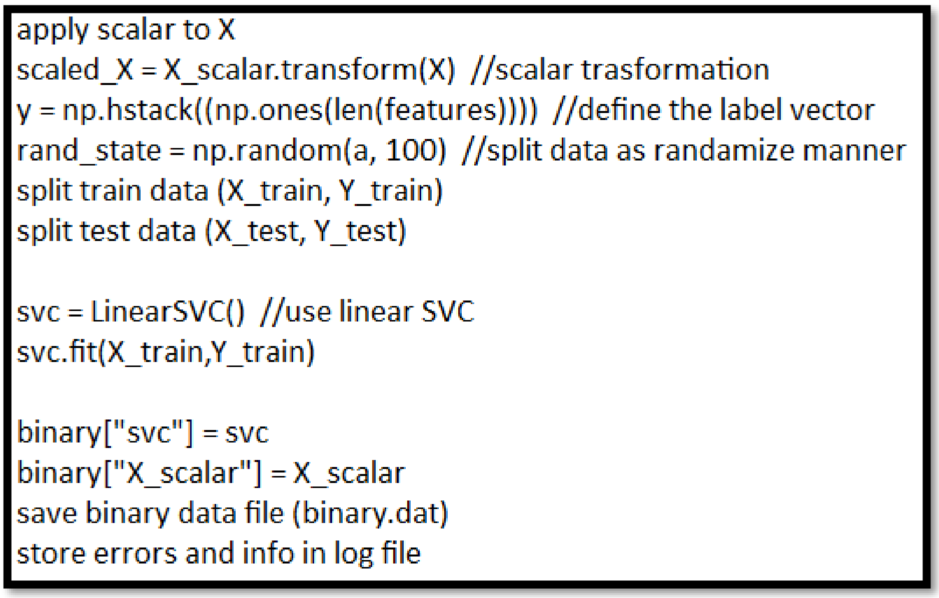


```
cnt /*count*/
R /*number of rows in image*/
C /*number of columns in image*/
x /*count the nuber of black pixels in each zone*/
px() /*one pixel*/
fv[] /*feature vector of a character*/

fv[0] = classification group
cnt = 1
i = 0
j = 0
while i < R
    while j < C
        x = 0
        for m=0 to 3
            for n=0 to 3
                if px(i+m, j+n)= black color
                    x++
                end if
            end for
        end for
        fv[cnt] = x
        cnt++
    j = j + 4
    end while
i = i + 4
end while
```

Figure 5.13: Algorithm of create the 65-dimentional Feature Vector

5.6 Recognition

Linear Support Vector Machine (SVM) and feature vector of each character were used to train the characters. Trained data set was stored separately. 75 Sinhala characters were trained successfully. Each character feature should be fit to each column scalar to start the SVM training. Each scalar transformed and scaled the value. Label vector was defined after scaling. Then the data was split as randomize manner for training and testing. These data fits to SVM model. Trained binary data was saved as .dat file. This trained file was used to recognized Sinhala characters. Algorithm to trained data using SVM is shown in Figure 5.14.



```
apply scalar to X
scaled_X = X_scalar.transform(X)  //scalar trasformation
y = np.hstack((np.ones(len(features))))  //define the label vector
rand_state = np.random(a, 100)  //split data as randamize manner
split train data (X_train, Y_train)
split test data (X_test, Y_test)

svc = LinearSVC()  //use linear SVC
svc.fit(X_train,Y_train)

binary["svc"] = svc
binary["X_scalar"] = X_scalar
save binary data file (binary.dat)
store errors and info in log file
```

Figure 5.14: Algorithm to trained data using SVM

Each Sinhala character was recognized using trained SVM sub model. Test image goes through SVM model prediction. If prediction is correct, recognize Sinhala character. According to recognition, display relevant character Unicode.

5.7 Chapter Summary

This chapter mainly discussed algorithms of horizontal projection profile, connected pixel labelling and peak and valley point identification. Also, this discussed character classification algorithms, feature vector creation algorithm and SVM data trained algorithm.

Chapter 6
Evaluation

6.1 Chapter Introduction

This chapter shows the result of each module and the calculated accuracy of overlapping and touching Sinhala handwritten character segmentation methods.

6.2 Result of Preprocessing

Preprocessing was done using noise removing, binarization and set black frame around the image. Figure 6.1 is shown the image after applying the preprocessing techniques.

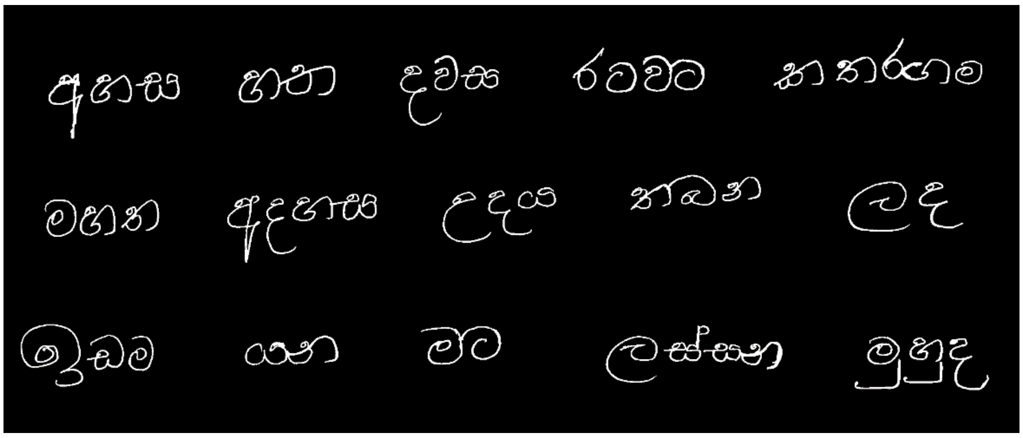


Figure 6.1: Image after applying the Preprocessing Techniques

6.3 Result of Line Segmentation

After preprocessing, line segmentation was done using horizontal projection profile. Figure 6.2 is shown the segmented first line.

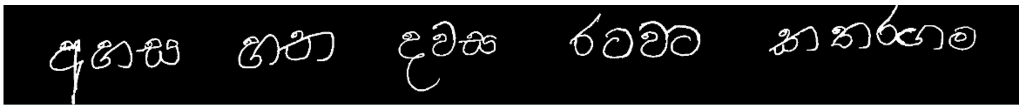


Figure 6.2: Segmented First Line

6.4 Result of Overlapping Character Segmentation

In segmented line, overlapping and isolated characters were segmented using connected pixel labelling method. Figure 6.3 is shown the segmented fourth character of the line.

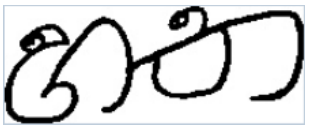


Figure 6.3: Segmented Fourth character of the line

6.5 Result of Touching Character Segmentation

After overlapping character segmentation, each character was scanned to find the touching points using peak and valley point identification method. If touching points were greater than zero, character was touched. Otherwise, character was isolated. Figure 6.4 is shown the segmented touched characters.

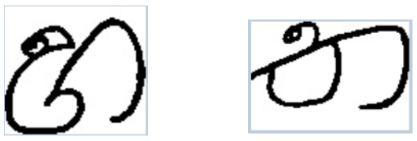


Figure 6.4: Segmented Touched characters

6.6 Evaluation of Overlapping and Touching Character Segmentation

Both segmentation algorithms were tested for 200 images. Understanding and measure the relevance was identified by the precision and recall fractions of test data [29]. Precision is shown in Equation (6.1). Recall is shown in Equation (6.2). Accuracy of the method is shown in Equation (6.3). tp, tn, fp and fn are called true positive, true negative, false positive and false negative respectively.

$$\text{Precision} = \frac{tp}{tp + fp} \qquad (6.1)$$

$$\text{Recall} = \frac{tp}{tp + fn} \qquad (6.2)$$

$$\text{Accuracy} = \frac{tp + tn}{tp + tn + fp + fn} \qquad (6.3)$$

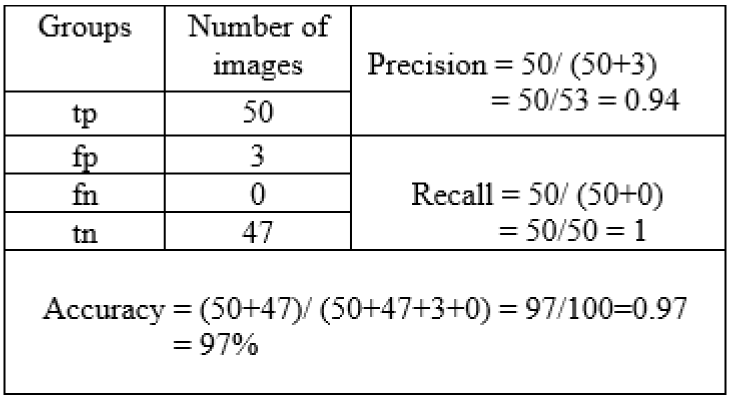
To measure the precision and recall of overlapping segmentation method, 50 overlapped images and 50 non- overlapped images were taken. Four conditions have been tested to calculate precision and recall. They are overlapped as overlapped(tp), non-overlapped as overlapped(fp), overlapped as non-overlapped(fn) and non-overlapped as non-overlapped(tn). Table 6.1 is shown the results of overlapping character segmentation.

To measure the precision and recall of touching segmentation method, 50 touched images and 50 isolated images were taken. Four conditions have been tested to calculate precision and recall. They are touched as touched(tp), isolated as touched(fp), touched as isolated(fn) and isolated as isolated(tn). Table 6.2 is shown the results of touching character segmentation.

In tested results, overlapping character segmentation (connected pixel labelling) has returned high precision and recall than touching character segmentation (peak and valley point identification). Connected pixel labelling method accuracy is 94% as shown in Table 6.1 and peak and valley point identification accuracy is 72% as shown in Table 6.2. According to [29] connected pixel labelling method returned most relevant results and peak and valley point identification method returned more relevant results than irrelevant results.

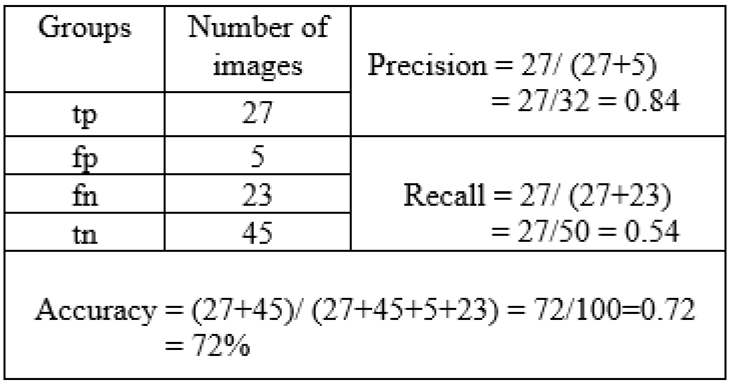
In touching character segmentation, if characters are touched at two points vertically, then it could not be segmented from this method.

Table 6.1: Results of Overlapping Character Segmentation



Groups	Number of images	
tp	50	Precision = 50/ (50+3) = 50/53 = 0.94
fp	3	Recall = 50/ (50+0) = 50/50 = 1
fn	0	
tn	47	
Accuracy = (50+47)/ (50+47+3+0) = 97/100=0.97 = 97%		

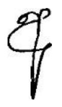
Table 6.2: Results of Touching Character Segmentation



Groups	Number of images	
tp	27	Precision = 27/ (27+5) = 27/32 = 0.84
fp	5	Recall = 27/ (27+23) = 27/50 = 0.54
fn	23	
tn	45	
Accuracy = (27+45)/ (27+45+5+23) = 72/100=0.72 = 72%		

6.7 Result of Classification and Feature Extractions

Example of character “ඇ” classified into descending character group. That character with its feature vector is shown in Figure 6.5. First element of that feature vector is the classification group. Descending character represents “3”.



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

Figure 6.5: A character with its feature vector

6.8 Results of Character Recognition

Sinhala handwritten character document was uploaded to the system. It is shown in Figure 6.6. Processed output is shown in Figure 6.7.

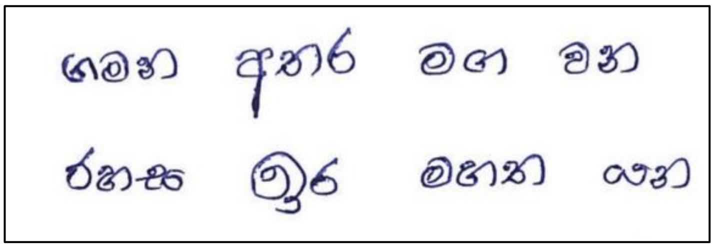
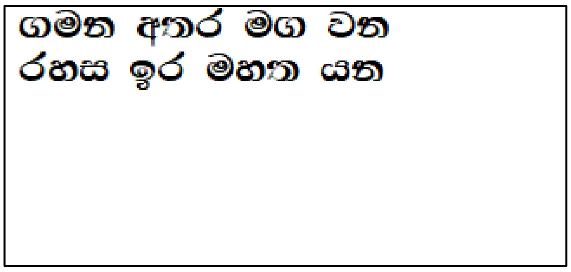


Figure 6.6: Uploaded Sinhala Handwritten Character document



ගමන අතර මග වන
රහස ඉර මහත යන

Figure 6.7: Processed Output

6.9 Chapter Summary

This chapter discussed results of each modules and precision, recall and accuracy of each segmentation methods.

Chapter 7
Conclusion and Future work

7.1 Chapter Introduction

This chapter briefly describes whole research work and all achievements of this research and future works.

7.2 Conclusion

Recently recognition of Sinhala characters using a computer is a popular problem in the research field. Some of the researches have been recognized machine-printed Sinhala characters [3], [4]. Sinhala handwritten character recognition can be a difficult task, if the characters are overlapped or touched.

This research had six sections. (image acquisition, preprocessing, segmentation, classification, feature extraction and recognition) Segmentation was main section of this research. Segmentation has two parts. (line segmentation and character segmentation) Line segmentation was done using horizontal projection profile.

Beginning of this research, character segmentation was divided into two parts. (overlapping character segmentation and touching character segmentation) The research was able to successfully segmentation of overlapping and touching Sinhala handwritten characters. According to the result, connected component labelling method is best method to segmentation of any overlapping character. Vertically two point touched character could not be segmented by peak and valley point identification method.

Feature vectors were created for each character using a classification group with 64 zones. Finally, Sinhala handwritten characters were recognized using linear SVM.

7.3 Future Work

As a future work, vertically two point touched characters will be segmented and improved the performance of touching character segmentation.

7.4 Chapter Summary

This chapter discussed the conclusion of this research and future works.

References

[1] Wikipedia, “Sinhalese language.” [Online]. Available: https://en.wikipedia.org/wiki/Sinhalese_language. [Accessed: 22-Oct-2017].

[2] S. Hewavitharana and N. D. Kodikara, “A Statistical Approach to Sinhala Handwriting Recognition,” no. November, 2015.

[3] H. L. Premaratne and J. Bigun, “A segmentation-free approach to recognise printed Sinhala script using linear symmetry,” Pattern Recognit., vol. 37, no. 10, pp. 2081–2089, 2004.

[4] M. Rimas, R. P. Thilakumara, and P. Koswatta, “Optical character recognition for Sinhala language,” c2013 IEEE Glob. Humanit. Technol. Conf. South Asia Satell. GHTC-SAS 2013, pp. 149–153, 2013.

[5] C. Silva and C. Kariyawasam, “Segmenting Sinhala Handwritten Characters,” vol. 2, no. January, pp. 22–26, 2014.

[6] M. L. M. Karunanayaka, N. D. Kodikara, G. D. S. P. Wimalaratne, R. Avenue, and S. Lanka, “Off Line Sinhala Handwriting Recognition with an Application for Postal City Name Recognition,” no. 35.

[7] S. Hewavitharana, H. C. Fernando, N. D. Kodikara, and S. Lanka, “Off-line Sinhala Handwriting Recognition using Hidden Markov Models,” 2002.

[8] B. Thakral and M. Kumar, “Devanagari handwritten text segmentation for overlapping and conjunct characters- A proficient technique,” Proc. - 2014 3rd Int. Conf. Reliab. Infocom Technol. Optim. Trends Futur. Dir. ICRITO 2014, 2015.

[9] R. J. Kannan, R. Prabhakar, and R. M. Suresh, “Off-line cursive handwritten Tamil character recognition,” Proc. - 2008 Int. Conf. Secur. Technol. SecTech 2008, vol. 4, no. 6, pp. 159–164, 2008.

[10] U. Sinha, “Connected Component Labelling,” 2010. .

[11] U. Pal, A. Belaid, and C. Choisy, “Water reservoir based approach for touching numeral segmentation,” Proc. Int. Conf. Doc. Anal. Recognition, ICDAR, vol.

2001–Janua, pp. 892–896, 2001.

[12] A. M. Zeki, “The segmentation problem in arabic character recognition the state of the art,” Proc. 1st Int. Conf. Inf. Commun. Technol. ICICT 2005, vol. 2005, pp. 11–26, 2005.

[13] L. R. K.A.K.N.D. Dharmapala, W.P.M.V. Wijesooriya, C.P. Chandrasekara, U.K.A.U. Rathnapriya, “Sinhala Handwriting Recognition Mechanism Using Zone Based Feature Extraction Key Terms :,” pp. 10–15, 2015.

[14] B. B. Chaudhuri, U. Pal, and M. Mitra, “Automatic recognition of printed Oriya script,” vol. 27, no. February, pp. 23–34, 2002.

[15] U. Pal and S. Datta, “Segmentation of Bangla unconstrained handwritten text,” Proc. Int. Conf. Doc. Anal. Recognition, ICDAR, vol. 2003–Janua, no. Icdar, pp. 1128–1132, 2003.

[16] F. Kurniawan, M. S. M. Rahim, D. Daman, A. Rehman, D. Mohamad, and S. M. Shamsuddin, “Region-based touched character segmentation in handwritten words,” Int. J. Innov. Comput. Inf. Control, vol. 7, no. 6, pp. 3107–3120, 2011.

[17] C. Scholl, “Recognition of Printed Sinhala Characters Using Linear Symmetry Recognition of Printed Sinhala Characters Using Linear Symmetry,” no. November 2002, 2013.

[18] H. L. Premaratne, E. Järpe, and J. Bigun, “Lexicon and hidden Markov model-based optimisation of the recognised Sinhala script,” Pattern Recognit. Lett., vol. 27, no. 6, pp. 696–705, 2006.

[19] P. Blunsom, “Hidden Markov Models,” Lect. notes, August, pp. 1–7, 2004.

[20] F. Jelinek, Statistical Methods for Speech Recognition. MIT-Press, 1998.

[21] LinkedIn Corporation, “Advantages and Disadvantages of Hidden Markov Model,” 2017. [Online]. Available: https://www.slideshare.net/joshiblog/advantages-and-disadvantages-of-hidden-markov-model.

[22] a. R. W. Rohana K. Rajapakse and E. K. Seneviratne, “a Neural Network

Based Character Recognition System for Sinhala Script," Vasa, no. January 1995, 1995.

[23] D. L. A. De Silva, "Sinhala OCR (Digital , Handwritten , & Palm-leaf Text) School of Computing , Asia Pacific Institute of Information Technology (APIIT), Sri Lanka ."

[24] O. Team, "OpenCV," © Copyright 2018, OpenCV team, 2018. [Online]. Available: https://opencv.org/. [Accessed: 10-Jul-2018].

[25] Microsoft Visual Studio - Wikipedia, "No Title," 2018. [Online]. Available: https://en.wikipedia.org/wiki/Microsoft_Visual_Studio. [Accessed: 10-Jul-2018].

[26] Wikipedia, "C++ - Wikipedia," 2018. [Online]. Available: https://en.wikipedia.org/wiki/C%2B%2B. [Accessed: 10-Jul-2018].

[27] K. S. A. Walawage and L. Ranathunga, "Segmentation of Overlapping and Touching Sinhala Handwritten Characters."

[28] Wikipedia, "Support-vector machine." [Online]. Available: https://en.wikipedia.org/wiki/Support-vector_machine.

[29] Wikipedia, "Precision and recall," 2018. .