Sinhala Handwriting Recognition Mechanism Using Zone Based Feature Extraction

K.A.K.N.D. Dharmapala[1], W.P.M.V. Wijesooriya[2], C.P. Chandrasekara[3], U.K.A.U. Rathnapriya[4], L. Ranathunga[5]
Department of Information Technology, University of Moratuwa
Katubedda, Sri Lanka
{naleendhanushka[1], vpowerrc[2], chinthakacccc[3], amdevex[4]}@gmail.com, lochandaka@uom.lk[5]

Abstract

Identification of Sinhala characters is considerably more difficult than other wide-spoken languages because of the complex shapes and similarities that are present within characters. With the addition of modifiers to the core characters, the recognition becomes increasingly more difficult. Most of the present systems only address the identification task of core characters which has potentially less real life applicability. The proposed solution tries to identify characters with or without touching and non-touching modifiers which can be effectively used in multiple applications.

Key Terms:

Sinhala, Handwriting Recognition, Preprocessing, Character Segmentation, Classification, Neural Networks, Image Processing.

1. INTRODUCTION

Sinhala language is currently more widely used in computers than it was a few years ago. Operating systems and computer applications provide support for Sinhala language interfaces and the conventional inputs through keyboard is more straightforward and hassle-free unlike the early days with the introduction and the widespread use of the Sinhala Unicode character scheme. Conversion of handwritten scripts to machine-recognizable characters remains as natural as interacting with a keyboard and real world applications vary from archiving existing handwritten documents to augmented reality based word translators [1].

In the Sinhala language, the character set is represented by 16 vowels, 2 semi-consonants, 40 consonants and 13 consonant modifiers. Modifiers are also known as strokes of characters [2]. These modifiers can be used in conjunction with consonants which can be positioned at different locations around the character. This addition of modifiers results in a very large number of possible shape combinations for an individual character symbol. Most of the current research in this area focuses only on a very limited number of characters, mostly without considering the character modifiers.

This limits the possible real world applicability. In this research we are trying to correctly identify characters with or without the presence of modifiers.

2. SINHALA HANDWRITING

When it's comes to Sinhala handwriting, touching and overlapping characters can be seen quite frequently, and automated segmentation of these types of characters can be a very complicated task. Unlike English, Sinhala characters have curves, which make it harder to find segmenting points. Apart from individual characters, there are character modifiers in the Sinhala language which need to be segmented too. Most of the time, modifiers and their related characters are touched or overlapped when written by hand.

Several research papers have been written on Sinhala character segmentation and recognition, but very few of them address the problem of segmentation of overlapping characters and characters with modifiers. Other papers are more focused on character classification rather than segmentation, and therefore they consider core characters which are not overlapping. But there are some suggested methods for touching and overlapping character segmentation which are quite complex and time consuming [3].

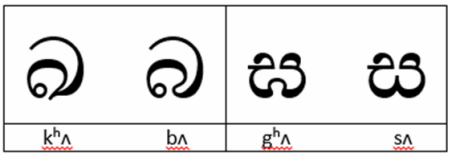

Figure 1: Different characters with similar shapes [2]

There are certain characters in the Sinhala alphabet which look mostly alike but which are actually different characters. Some examples are shown in Figure 1. Vast variations of writing styles of individuals also make it difficult to successfully implement a Sinhala character recognition and classification method. Figure 2 gives an example for writing variations.

Figure 2: Similar characters with different shapes

Additionally, there are certain modifiers which are very difficult to distinguish only based on their features. These characters have to be identified with respect to their context. Furthermore, different writing styles make it even more difficult to identify the characters based only on the context.

3. RECOGNITION SYSTEM

This section describes the implemented experimental handwriting recognition system in four major sections. The first section deals with the preprocessing which followed by segmentation. Feature extraction and classification are the other two sections that are explained.

3.1 Preprocessing

A4 sized white papers were used to collect sample handwriting. A single document typically includes 10 lines with

7-8 words in each line. The documents are scanned at a resolution of 300 dpi and stored as 8-bit grayscale images. Before segmenting the characters, the images should undergo a pre-processing stage in order to reduce noise and fix character alignment issues. Several methods and techniques are used in this stage such as contrast equalization, noise reduction, binarization and skew correction of text lines which are explained below.

3.1.1 Contrast Equalization

Contrast limited adaptive histogram equalization (CLAHE) [4] technique is used to equalize the intensities throughout the image. This will improve the accuracy of the binarization step. Unlike in traditional histogram equalization, the CLAHE technique creates separate histograms for multiple segments of the image which increases the accuracy of the equalization process.

3.1.2 Noise Reduction

This step is critical because accuracies of text segmentation and classification are directly dependent on it. There are mainly two types of noises to deal with which are called "Gaussian noise" and "Salt & Pepper noise". To minimize the Gaussian noise "Non-Local Means Denoising" [5] algorithm is used. Mean filter is used to deal with salt and pepper noise. Filtering is done by applying a 3×3 or similar filter mask. The mask is passed over the entire image and the value for the center pixel is calculated.

3.1.3 Binarization

Binarization is the technique of converting an image to binary format containing only 2 values. This can be achieved by applying a threshold function. In simple thresholding, a global threshold value will be used for the whole image. But it may produce bad results in certain conditions where image has different lighting conditions in different areas. The solution is to use “Adaptive Thresholding”. This algorithm calculates separate threshold values for different regions of the image and will give better results for images with varying illumination.

3.1.4 Skew Correction

When an image is captured it could be slightly angled. So it’s necessary to compute the skew angle and to rotate the text before further processing. There are several methods which can be used to detect the skew angle such as using "Probabilistic Hough Transform" [6]. It generates imaginary straight lines across the image according to the distribution of the black pixels which can be used to calculate the skew angle and rotate the image afterwards.

3.2 Segmentation

Character segmentation is done after pre-processing the image. The main objective of this module is to segment characters and their modifiers. The segmentation process is done using various image processing algorithms to distinguish the different letters and words in the input text sample. As the output of this module, segmented characters are generated as images which are given as inputs to the character classification module.

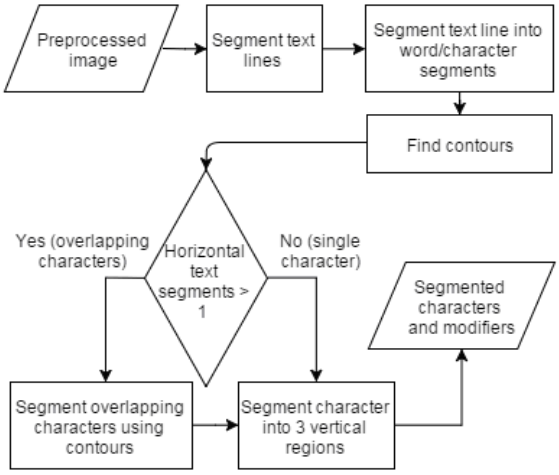

Figure 3: Character segmentation flow chart

3.2.1 Segment Text Lines

Segment Text Lines: Horizontal projection profile of the image is used to segment text lines. As seen in Fig. 4, gaps between the lines can be clearly identified using the horizontal projection profile of the entire image. Valleys of the graphs represent the horizontal gaps between text lines. In the projection profile, if there exist a run of at least a certain number of consecutive 0’s then the midpoint of that run is considered as the boundary of a text line. In order to ignore noise and touching segments between text lines, if the aggregation of the pixel values of a horizontal line is less than a certain value it will be replaced with 0. That way it is assured to detect accurate local valleys of the projection profile.

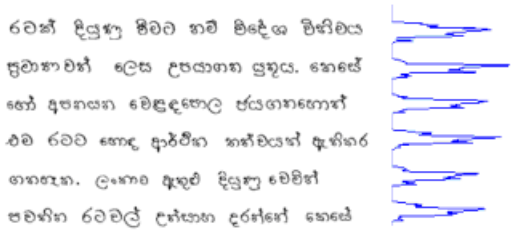

Figure 4: Horizontal projection profile

3.2.2 Segment Horizontal Characters

Vertical projection profile of a text line can be used as the base method of segmenting character, character modifiers and words. As shown in the Fig. 5, similar to identifying the text lines, gaps between the vertical lines can be clearly identified and can be used to identify boundaries of characters and words. But this technique can be only used when characters are separated clearly from each other. When there are overlapping and touching characters present, analyzing the vertical projection profile won’t be enough because there won’t be gaps between characters when they are connected or overlapped. Further processing is needed in order to segment these characters.

3.2.3 Count Number of Horizontal Text Segments

Contours are curves joining all the continuous points in the boundary having the same color or intensity. They are useful for shape analysis and object detection and recognition.

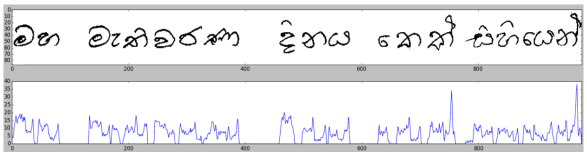

Figure 5: Vertical projection profile

Contours are identified from the binary image using the "findContours" function from OpenCV python library [7]. When counting the number of character segments, only external contours should be considered, else the same character will be counted multiple times. External contours only include "outer" contours, so for example; if one contour is enclosing another (as concentric circles), only the outermost is given. If the number of contour groups is more than one, it is safe to assume that the text segment contains overlapping characters.

3.2.4 Segment Overlapping Characters

Overlapping characters and their locations are identified in the previous step. But exact horizontal margins cannot be calculated for overlapping characters because they are overlapping with each other. The solution is to retrieve all the contours of each character and redraw the character using them on an empty image canvas. Contours should be retrieved with a two level hierarchical structure where the top level contains external boundaries of the characters and the bottom level contains boundaries of the holes inside characters. Combining all of those points and then filling the space between boundaries will construct the characters again in their original form. Using this method, overlapping characters can be separated and reconstructed separately (Fig 6.).

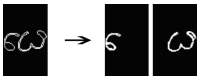

Figure 6: Segment characters using contours

3.2.5 Vertical Character Segmentation

In the character classification module, classification is done using three separate neural network models aimed at three zones of a character. Any given character lies well within upper, middle and lower zones defined for the input image for the character classification module. Some character units are written such that it is dispersed within all the three zones while some other character units may be written only in one or two of the zones. In most of the cases, the middle zone of any given character remains unchanged while most of the modifiers are being added to either the upper or lower zone [8] [9]. Table I shows typical examples of zonal representation of Sinhala characters.

In order to identify boundaries of each region vertical segmentation has to be done for each character. Characters are divided into three zones using two bases. Top and bottom character modifiers need to be segmented to top and bottom regions in order to classify them correctly. So base lines should be calculated accordingly.

In order to find the correct positions of the vertical segments, firstly, contours are used to find if there are any vertical modifiers attached to the character. If any are found, their

Table 1: CHARACTER UNITS IN ZONES

Character Unit	*Lower Zone*	*Middle Zone*	*Upper Zone*
	None		
			None
			None
			None
	None		
			None
	None		

positions are used to segment the character into vertical segments easily and accurately.

If there are no vertical modifiers found, several other methods have to be followed in order to find the segmentation positions. Firstly, the base line locations for the whole text line are calculated and kept as preliminary measures of baseline locations. It is found that positions of the top two highest horizontal pixel density lines are similar to the positions of the required base lines. “Argrelextrema” function from Scipy Python library is used to find all local maximas in the image [10]. From them, points which are lower than a certain value are ignored. Then the two points which are situated farthest from each other are selected as the base line locations (Fig 7.).

After calculating the base line locations for the whole text

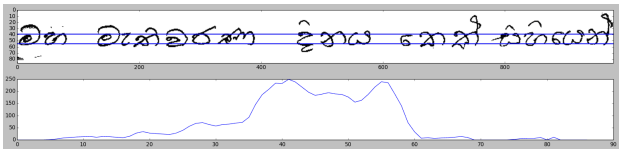

Figure 7: Identify the base lines of text lines

line, baseline locations for each character are calculated too because baseline locations calculated for a text line may not be the most accurate baseline locations when considering a single character. The same method will be used to calculate baseline locations in a single character. But previously calculated baseline locations will also be used in order to find the best possible locations for the baselines. Lower baseline calculated from the text line is correct for most of the scenarios. But calculating the upper baseline location will be tricky since there can be modifiers above the characters. In those cases the upper baseline should be calculated in such a way that any modifier falls above the baseline.

3.3 Feature Extraction

After the character segmentation process, to increase the classification accuracy, it is essential to choose the most suitable feature extraction method. Feature extraction is performed separately for the three zones as shown in Fig 8. The features that are extracted for any given zone are independent from the features of the other two zones of the character unit except for the width and height features.

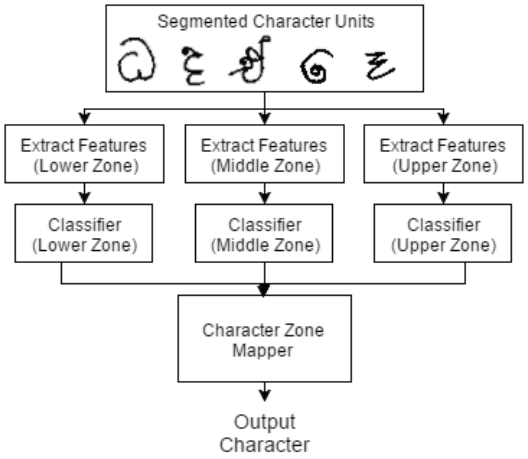

Figure 8: Feature Extraction and classification flowchart

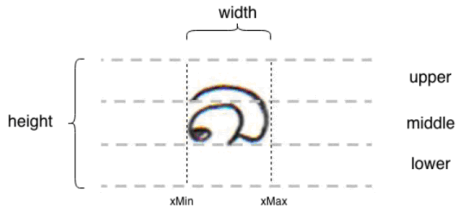

Figure 9: Character unit within the zones

Each zone is divided into two vertical blocks and their vertical projection histogram is taken as a feature vector. The horizontal projection of two horizontally divided blocks is also taken as a feature vector. These feature vectors are considered for all the three zones.

Additionally, the middle zone is resized and a sparse matrix representing the character's shape is considered as well. The image is resized using the nearest neighbour interpolation method.

3.4 Classification

Once the features are extracted separately for the three zones, those features are used to train three artificial neural networks with back propagation to perform the classification based on new inputs later on. Each of these neural networks only evaluates and classifies the respective zone to which it is assigned and they are not dependent on the features of the other zones other than certain features of the character as a whole (e.g. - height, width of the character as a whole). For this training process 30 samples from the data set are used, which includes the 48 characters without modifiers and a selected set of characters with modifiers that has unique upper and lower zone representations.

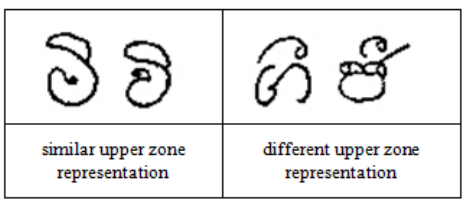

Figure 10: Zone representations of characters

Once the neural networks are trained they can classify new input features attributing them to the respective labels of upper, middle and lower zones. Neural networks for each of the zones provide an approximate probability for every input data set giving an idea on how much a given input feature set can be attributed to an existing label. This approximate probability is calculated by using the sigmoid transfer function of the output nodes which is mapped from [-1, 1] to [0, 1].

These approximate probability values are then used to calcu-

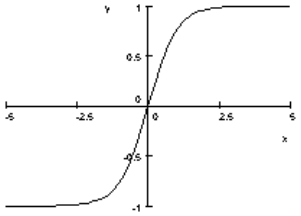

Figure 11: Sigmoid Error Function

late the joint probabilities given that P(L), P(M) and P(U) are independent of each other [11] for every possible combination of values.

P(U) : probability of getting the label "u" for upper zone
P(L) : probability of getting the label "l" for lower zone
P(M) : probability of getting the label "m" for middle zone
Given that the probabilities of each zone are independent;

$$P(L, M, U) = P(L) \times P(M) \times P(U) \quad (1)$$

P(L,M,U) values are calculated for all the combinations of lower, upper and middle zones along with their joint probability values. The joint probability values are then matched against a map of all the possible character combinations starting from the highest joint probability value and then proceeding to lower joint probability values if a match is not found in the map at higher values. This map of possible zone combinations will avoid the chances of generating a character unit which is not present in the Sinhala alphabet. For example, the probability values for different zones may return a very high joint probability values for a character like which is not valid in the Sinhala language. In such a case the character map would consider the next highest joint probability value because that character unit is not available in the character zone map.

4. EXPERIMENTAL RESULTS

Accuracy of the segmentation can be measured in different stages such as line segmentation, single character segmentation, overlapping character segmentation and vertical character segmentation. The dataset which was used in the testing stage contained 12 text lines written by 35 different people. It doesn't contain touching characters or touching horizontal modifiers, but it contains overlapping characters and touching vertical modifiers. Furthermore, text lines in the dataset are well separated from each other without overlapping or touching vertically. Table II given below shows the test results obtained.

The classification accuracy is evaluated for each neural network of each zone. The classification accuracy is the number of correct predictions made divided by the total number of predictions which is then multiplied by 100 to take a percentage value. This gives a numeric measurement on

Table 2: SEGMENTATION ACCURACIES

Stage	Accuracy
Text line segmentation	100%
Single character segmentation	98%
Overlapping character segmentation	95%
Vertical character segmentation	84%

how the classification accuracy changes for different zones. To calculate the classification accuracy 10 samples are used which were not used in the training set. For each of these zones there might be one or more characters that have the same lower, middle, or upper character representation. For example, "ඵ" and "ඵ" has the same middle zone character representations.

Table 3: CLASSIFICATION ACCURACIES FOR DIFFERENT ZONES

Zone	Classification Accuracy
Lower	0.746
Middle	0.661
Upper	0.744

The final output character unit is generated only after the best matching character unit is selected from the character zone map. The accuracy of correctly mapping the zones to matching character unit is calculated by the same method of calculating the classification accuracy where the correctly classified character unit instances are divided by the total number of character unit instances tested. To calculate the output character accuracy, 10 samples are chosen from the initially collected dataset which were not already used in the training set.

Table 4: CLASSIFICATION ACCURACIES FOR OUTPUT CHARACTERS

Expected result	*Classification accuracy*	*Predicted result*
ට	1.0	ට, ට, ට, ට, ට, ට, ට, ට, ට, ට
ඩ	0.6	ඩ, ඩ, ඩ, ඹ, ඩ, ඩ, ඹ, ඩ, ඹ, ඩ
ඨ	0.9	ඨ, ඨ, ඨ, ඨ, ඨ, ඨ, ඨ, ඨ, ඪ, ඨ
ර	0.9	ර, ර, ර, ර, ර, ර, ඊ, ර, ර, ර
ය	0.9	ඝ, ය, ය, ය, ය, ය, ය, ය, ය, ය
ල	0.9	ල, ල, ල, ල, ල, ල, ළ, ල, ල, ල
ම	0.4	ව, ම, ඹ, ම, ම, ම, ට, ව, ට, ඔ
මි	0.8	මි, ධි, මි, මි, මි, මි, මි, මි, වි, මි
මි	0.6	මි, ධි, මි, මි, ඕ, මි, ට, මි, ධි, මි
මි	0.5	ඊ, ඩි, ඒ, ඕ, මි, මි, මි, මි, ඒ, මි
කු	0.4	ඤු, කු, ඤු, කු, කු, ඛු, ඤු, කු, තු, තු
ක	0.5	ක, ක, ඟ, ක, ක, ක්ය, න, න, ක, ක
ත	0.5	ත, ත, ඤ, ත, ත, ත, ත, ත, ත, ත
තු	0.1	ඤු, ගු, තු, තු, තු, තු, තු, ඤු, ඟු, තු

5. CONCLUSION

In this paper, a system to classify Sinhala Handwritten Characters is developed using a pre-classification approach. First, the text document is preprocessed using several methods such as contrast equalization, noise reduction, binarization, skew correction of text lines, etc. Then the text lines and characters and horizontal character modifiers are segmented using methods such as horizontal/vertical projection profile analysis and contour analysis for identifying overlapping characters. Afterwards, each character is segmented into three vertical zones as upper, lower and middle using contour analysis and vertical projection profile analysis.

After the segmentation process the feature extraction is performed on the three zones separately to extract the relevant features. The classification is also performed separately on the three zones to classify the zones attributing them to respective character units with probability values measuring how relevant a prediction is to the real character. Then the calculated joint probability is used to select the best matching character from a map of possible character unit combinations which contains the full Sinhala alphabet.

The main issue faced in the segmentation process is the abnormal writing styles, which are not according to the standard Sinhala writing style. This makes it harder to do the vertical segmentation accurately. For example; most people would write letters such as "ම", "ව", "ට", "ඩ", "ඨ" etc. smaller compared to other letters. Originally, the top part of these letters should be segmented as the upper zone, but since letters are written smaller they would be wrongly segmented into the middle zone. Another issue is handling touching characters, which is not yet resolved in this system due to the complexity in Sinhala letters. However, touching vertical character modifiers are segmented to a certain extent.

From the classification test results it can be noticed that there are confusions between characters like "කූ" and "කූ" . Their middle zone representation is dense with features that make it difficult to identify it from characters like "ඤ", "ඟු" and"තූ" . Additionally, lower character modifiers in cases like "ප්‍ර", "ප්‍ව", "ප්‍ල" are difficult to classify accurately due to the similarity of the modifiers and the individual writing styles of those modifiers. The classification accuracies are considerably better in cases like "කූ", "තූ", where the upper character modifiers have more signifying features than the lower character modifiers. Characters like "මි", "මි", "මි", have very high classification accuracies because their lower, middle and upper zone features are considerably different to the other characters.

Currently this system cannot segment characters which are touching horizontally, although it can segment touching vertical character modifiers. That feature can be added as an extension to this system. Furthermore spelling correction module can be used to increase the accuracy of the predicted words and sentences. In addition to that a template based recognition method to identify the characters based on the basic character symbols is under investigation to avoid the discrepancies that occur due to abnormal writing [12]. This approach can be highly recommended for optical character recognition where the printed characters would be written strictly adhering to the three zone framing.

References

[1] Tabish Khan, Rishisingh Hora, Ashwin Bendre, Prof. Sneha Tirth, “Augmented Reality Based Word Translator”, International Journal of Innovative Research in Computer Science & Technology (IJIRCST), vol. 2, no. 2, 2014.

[2] M. A. P. Chamikara, S. R. Kodituwakku, A. A. C. A. Jayathilake, K. R. Wijeweera, “Fuzzy Neural Hybrid method for Sinhala Character Recognition”, International Journal of Advanced Research in Computer Science and Software Engineering, vol. 4, no. 9, 2014.

[3] M.L.M Karunanayaka, N.D Kodikara, G.D.S.P Wimalaratne, "Off Line Sinhala Handwriting Recognition with an Application for Postal City Name Recognition", University of Colombo School of Computing, 6th International Information Technology Conference on From Research to Reality, pp. 23-29, 2004.

[4] Zuiderveld, K., "Contrast Limited Adaptive Histogram Equalization", Chapter VIII.5, Graphics Gems IV, Cambridge, MA, Academic Press, pp. 474-485, 1994.

[5] A. Buades, B. Coll and J. Morel, "Non-Local Means Denoising", Image Processing On Line, vol. 1, 2011.

[6] N. Kiryati, Y. Eldar, A.M. Bruckstein, "A Probabilistic Hough Transform", Pattern Recognition, vol. 24, issue 4, 1991, pp. 303-316.

[7] Suzuki, S. and Abe, K., "Topological Structural Analysis of Digitized Binary Images by Border Following", Computer Vision, Graphics, and Image Processing, vol. 1, no. 30, pp. 32-46, 1985.

[8] S. Hewavitharana, Dr. N.D. Kodikara "A Statistical Approach to Sinhala Handwriting Recognition", International Conference on Advances in ICT for Emerging, 2015.

[9] B. Jayasekara and L. Udawatta, “Non-Cursive Sinhala Handwritten Script Recognition: A Genetic Algorithm Based Alphabet Training Approach”, Proc. of the International Conference on Information and Automation ICIA2005, Colombo, Sri Lanka, pp. 292-297, 2005.

[10] P. Du, W. Kibbe and S. Lin, ’Improved peak detection in mass spectrum by incorporating continuous wavelet transform-based pattern matching’, Bioinformatics, vol. 22, no. 17, pp. 2059-2065, 2006.

[11] Encyclopediaofmath.org, ’Joint distribution - Encyclopedia of Mathematics’, 2015. [Online]. Available: https://www.encyclopediaofmath.org/index.php/Joint_distribution. [Accessed: 31- Oct- 2015].

[12] Mo Wenying, Ding Zuchun Guangdong “A Digital Character Recognition Algorithm Based on the Template Weighted Match Degree”, ASTL: Advanced Researches on Computer and Applications, vol. 17, pp. 53-60, 2013.