PERGAMON

Available at
www.ElsevierComputerScience.com
POWERED BY SCIENCE DIRECT®

Pattern Recognition 37 (2004) 2081–2089

PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY

www.elsevier.com/locate/patcog

A segmentation-free approach to recognise printed Sinhala script using linear symmetry

H.L. Premaratne[a,b,*], J. Bigun[a]

[a] School of Information Science, Computer and Electrical Engineering, Halmstad University, Box 823, Halmstad S-301 18, Sweden
[b] University of Colombo School of Computing, 35, Reid Avenue, Colombo 07, Sri Lanka

Received 5 June 2003; accepted 30 January 2004

Abstract

In this paper, a novel approach for printed character recognition using linear symmetry is proposed. When the conventional character recognition methods such as the artificial neural network based techniques are used to recognise Brahmi Sinhala script, segmentation of modified characters into modifier symbols and basic characters is a necessity but a complex issue. The large size of the character set makes the whole recognition process even more complex. In contrast, in the proposed method, the orientation features are effectively used to recognise characters directly using a standard alphabet as the basis without the need for segmentation into basic components. The edge detection algorithm using linear symmetry recognises vertical modifiers. The linear symmetry principle is also used to determine the skew angle. Experiments with the aim for an optical character recognition system for the printed Sinhala script show favourable results.

Keywords: Optical character recognition; Linear symmetry; Dynamic threshold; Skew correction

1. Introduction

Research on the development of OCR systems for Brahmi descended South Asian scripts have been on the records since early 1970s. But, only a very few of these languages possess commercial OCR products at present, indicating the challenges faced by reseachers. The very nature of South Asian scripts has proved that the methodologies used in developing OCR systems for Latin scripts such as English cannot be applied directly to these alphabets. One reason for this is the complexity experienced in the use of various modifier symbols to modify consonants in generating different vocal sounds in South Asian scripts which can be compared to the combining of consonants and vowel letters in the Latin scripts [1]. The segmentation of typed South Asian script is not trivial as in the case of typed Latin script consisting only around 30 symbols in the alphabet.

A careful study shows that the avenues for applying the methods that have been proposed to recognise other South Asian Alphabets such as Devanagari, Telugu [2] to Sinhala are slim due to the considerable differences that exist between Sinhala and other South Asian scripts. Lack of some prominent features such as strokes, segments of straight lines and junctions in Sinhala characters that are common in many other South Asian scripts [3,4], is one example. This is explained by the generally rounded shape of characters in the Sinhala alphabet. Therefore, the necessity arises for a suitable approach that could explore the unique features of Sinhala characters in the process of recognition of the Sinhala script. This paper proposes the effective use of orientation features of the Sinhala script in skew correction and recognition. Further, the proposed algorithm does not require the segmentation of the script into basic components which could have been a highly complex issue.

Results of the experiments show the accuracy rate as high as in many works carried out for other South Asian alphabets.

* Corresponding author. Tel.: +94-11-2589123; fax: +94-11-2587239.

E-mail addresses: hlp@ucsc.cmb.ac.lk (H.L. Premaratne), josef.bigun@ide.hh.se (J. Bigun).

0031-3203/$30.00 © 2004 Pattern Recognition Society. Published by Elsevier Ltd. All rights reserved.
doi:10.1016/j.patcog.2004.01.021

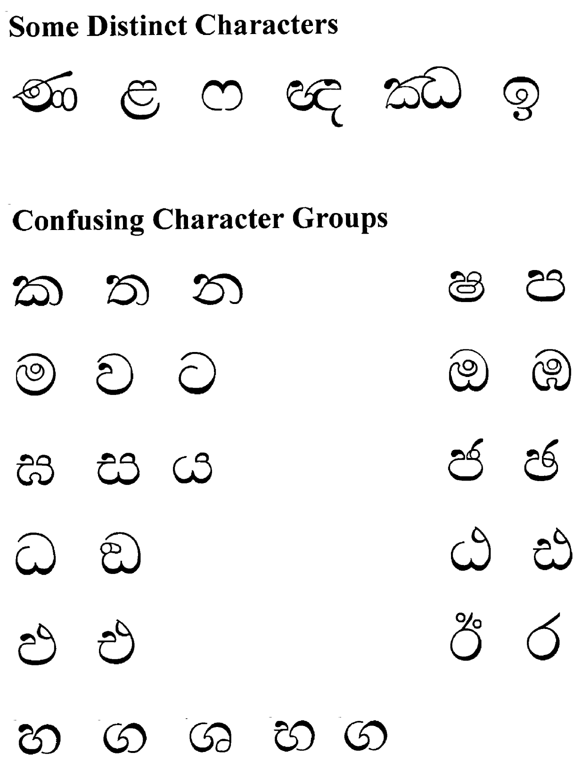

Fig. 1. Distinct and similar shapes of characters.

1.1. *Alphabet and the modification process*

The Sinhala script used by over 80% of the 18.5 million population in Sri Lanka has been descended from the ancient Brahmi script and evolved independently over two millennia many centuries. The Sinhala language is unique to Sri Lanka and the Sinhala characters that are generally round in shape differ from all the other Brahmi descended scripts in South Asia. The Sinhala alphabet consists of 18 vowels, 41 consonants and 17 modifier symbols. A vowel may appear only as the first character of a word and a consonant is modified using one or more of the modifier symbols to produce the required vocal sound. The total number of different modifications from the entire alphabet including the basic characters is nearly 400. Although each character possesses a distinct characteristic shape to distinguish from the others, some characters resemble with one or more of the other characters by their appearance. Some examples are given in Fig. 1.

Modification of a character is carried out by simply adding one or more modifier symbols before/after/above/below the character without affecting its general shape. However this rule is violated for a specific subset of the alphabet numbering to 10 characters, in most of the printed scripts, to give a better appearance (Fig. 2). Also, in some modifications, the joint between the character and the modifier symbol is smoothed to make the modified character appear as a single unit of symbol (see Fig. 3).

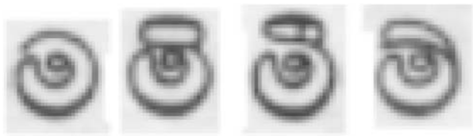

Fig. 2. Violation of modification rule by changing the shape of a character (basic character is in the extreme left).

1.2. *Characteristics of the script*

A single line of script is organised in three horizontal layers. The middle layer contributing to approximately 50% of the total line height, mainly include 15 basic characters and 9 modifier symbols. Twenty two other basic characters occupy the middle layer and the upper layer, with approximately 75% and 25% of the total height of each character in each layer, respectively. The middle and the lower layers include the remaining 8 characters, with approximately 75% and 25% of the total height of each character in each layer, respectively. Four modifiers occupy the upper layer while the remaining 5 modifiers are assigned to the lower layer. The upper and the lower layers are of equal height each having 25% of the total line height (Fig. 4).

2. Recognition process

2.1. *Theory*

The theory used in the recognition process is the orientation field tensor [5,6] which has been used effectively in many applications over the past few years. A local neighbourhood with ideal local orientation is characterised by the fact that the grey value only changes in one direction. In all other directions it is constant. Since the grey values are constant along lines, local orientation is also denoted as linear symmetry [5]. The linear symmetry is also represented in the form a vector. Since the direction of a simple neighbourhood is different from the direction of a gradient, which is strictly cyclic, representation of the linear symmetry needs the doubling of the angle of orientation. The vector that represents the linear symmetry is composed of two quantities. One is the orientation angle and the other is the certainty measure.

2.1.1. *Mathematical representation*

The local orientation is determined using the following three steps [6].

(i) Select a local neighbourhood from the image using a window function.

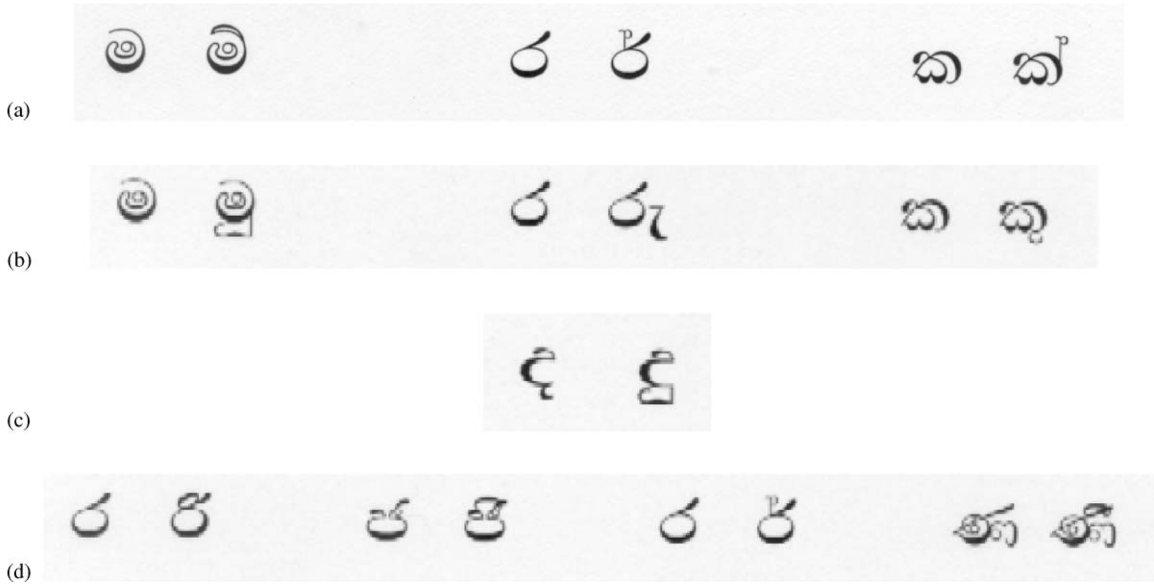

Fig. 3. Same phonetic sound but (a) two different modifier symbols, (b) three different modifier symbols, (c) modification by changing the original shape of character, (d) modifier symbol placed inside the frame of the basic character.

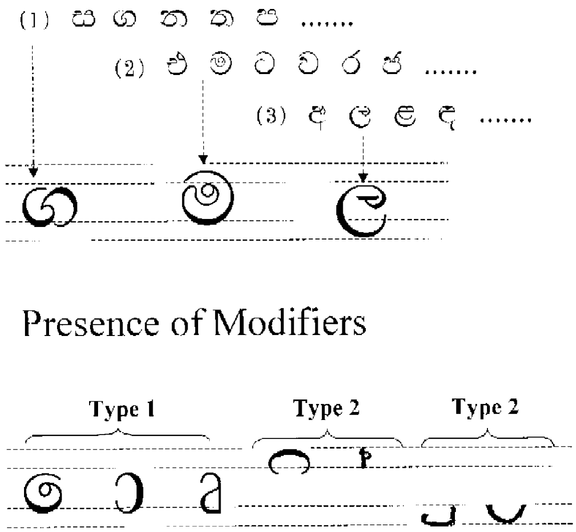

Fig. 4. Three-layered structure of a line.

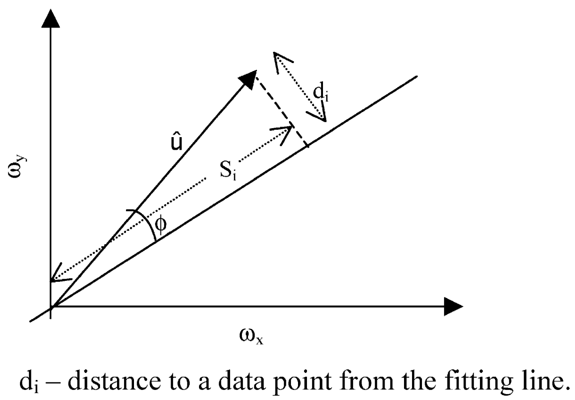

Fig. 5. Representation of linear symmetry in the frequency domain.

(ii) Fourier transforms the windowed image.
(iii) Determine the local orientation by fitting a straight line to the spectral density distribution.

When fitting a straight line, the sum of the squares of the distances of the data points are minimised (see Fig. 5).

Since the minimisation of d_i is the same as the maximisation of S_i, the following equation is obtained:

$$\min_{\phi} \iint d_i^2(\omega)\, \mathrm{d}\omega_x\, \mathrm{d}\omega_y$$

$$= \max_{\phi} \iint S_i^2(\omega)\, \mathrm{d}\omega_x\, \mathrm{d}\omega_y$$

$$= \max_{\|\mathbf{u}\|=1} \mathbf{u}^{\mathrm{T}} \underbrace{\left(\iint \begin{pmatrix} \omega_x^2 & \omega_x\omega_y \\ \omega_x\omega_y & \omega_y^2 \end{pmatrix} \mathrm{d}\omega_x\, \mathrm{d}\omega_y \right)}_{J} \mathbf{u}.$$

The orientation is obtained as the eigenvector of the largest eigenvalue of structure tensor J. J can be rotated so that it is diagonalised. The rotation matrix is in fact the eigenvector matrix,

$$\begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix} = \underbrace{\begin{pmatrix} \cos\phi & \sin\phi \\ -\sin\phi & \cos\phi \end{pmatrix}}_{\mathbf{u}^{\mathrm{T}}} \times \underbrace{\begin{pmatrix} J_{xx} & J_{xy} \\ J_{xy} & J_{yy} \end{pmatrix}}_{J} \underbrace{\begin{pmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{pmatrix}}_{\mathbf{u}}.$$

Comparison of the diagonal elements on both sides of the equation gives

$$\lambda_1 + \lambda_2 = J_{xx} + J_{yy},$$

$$\begin{aligned} \lambda_1 - \lambda_2 &= (J_{xx} - J_{yy})\cos 2\phi + 2J_{xy}\sin 2\phi \\ &= (J_{xx} - J_{yy}, +2J_{xy}) \begin{pmatrix} \cos 2\phi \\ \sin 2\phi \end{pmatrix} \\ &= \left\langle I_{20}, \begin{pmatrix} \cos 2\phi \\ \sin 2\phi \end{pmatrix} \right\rangle \\ &= \langle I_{20}, \|I_{20}\|/\|I_{20}\| \rangle = I_{20}. \end{aligned}$$

Therefore, $\lambda_1 - \lambda_2 = \|I_{20}\|$.

Define $\nabla f = \partial f/\partial x + \mathrm{i}(\partial f/\partial y)$, then

$$\begin{aligned} I_{20} &= \int (\nabla f)^2 = \int ((\partial f/\partial x)^2 - (\partial f/\partial y)^2 \\ &\quad + 2\mathrm{i}(\partial f/\partial x)(\partial f/\partial y)) \\ &= \int [(\omega_x + \mathrm{i}\omega_y)^2(\omega_x - \mathrm{i}\omega_y)^0|F|^2] \\ &= (\lambda_1 - \lambda_2)\exp(2\mathrm{i}\phi), \end{aligned}$$

$$\begin{aligned} I_{11} &= \int [(\omega_x + \mathrm{i}\omega_y)^1(\omega_x - \mathrm{i}\omega_y)^1|F|^2] \\ &= \int (\omega_x^2 + \omega_y^2)|F|^2 \\ &= \int (\partial f/\partial x)^2 + (\partial f/\partial y)^2 = \lambda_1 + \lambda_2. \end{aligned}$$

Angle of I_{20} represents the ($2 \times$ angle) where the angle is the inclination angle of the fitting orientation if the linear symmetry exists, and I_{11} represents the sum of the best and the worst total errors.

The linear symmetry algorithm that extracts the tensor is characterised by the fact that it delivers a dense orientation field along with certainties. In case of high confidence on the existence of orientation, the linear orientation represents the least change of grey values in one direction and maximal change in the orthogonal direction. Hence a linear symmetry tensor for an image is constructed by averaging the orientation of the local neighbourhood, for each pixel of the image (see Fig. 6).

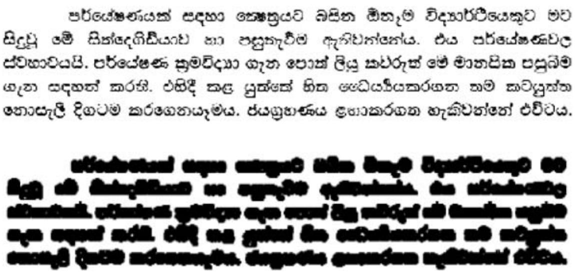

Fig. 6. Original image (above) and linear symmetry tensor (below).

2.1.2. Implementation

In the implementation, the linear symmetry (LS) tensor of the image of the script to be recognised, is initially built. Each pixel of this LS tensor (i.e. I_{20} defined in Section 2.1.1) which is represented as a vector will be of the form $x + jy$ where $j = \sqrt{(-1)}$. The length of the vector is a measure of the local 'LS strength' and the argument is the estimated local orientation.

The I_{20} vector is constructed by filtering the original image with two derivative filters (created as Gaussian kernels), in the x-direction and in the y-direction, respectively, and by combining the two resulting images ($\mathrm{d}xf$ and $\mathrm{d}yf$) as $I_{20} = (\mathrm{d}xf + j^*\mathrm{d}yf)^2$.

(In the MATLAB® implementation, four Gaussian kernels $\mathrm{d}x$, $\mathrm{d}y$, gx and gy are created and used as derivative filters [5], to filter the image using the Matlab function 'filter2' which does a 2-D convolution. The resulting images $\mathrm{d}xf$ and $\mathrm{d}yf$ are obtained by the filtering operations

filter2(gy, filter2(dx, img)) and filter2(gx, filter2(dy, img)), respectively, where img is the original image representing a script).

In the process of identification of a certain Sinhala character, the following principle is used:

$$\max_{|b|=1} (b^*)^t a = \gamma |a|^2,$$

where b is of the form $a + je$, represents the LS tensor of the character and a is of the form $\bar{a} + j\bar{e}$, represents a frame within the LS tensor (I_{20}) of the image of the script. The dimensions of the frames of a and b are the same. $a, e, \bar{a}$ and $\bar{e}$ are real values and γ is an arbitrary constant.

In other words, the scalar product $(b^*)^t a$ (referred to as 'correlation' in the future) is maximum if the elements of b are in the complex conjugate direction of the elements of a. Therefore, when the LS tensor of the character which is being recognised coincides with an occurrence of the same character in the LS tensor of the image, the product $(b^*)^t a$ will attain a relatively high value. (*Note*: b, which is the LS tensor of the character being recognised is extracted in advance from the LS tensor of an image of a similar script.)

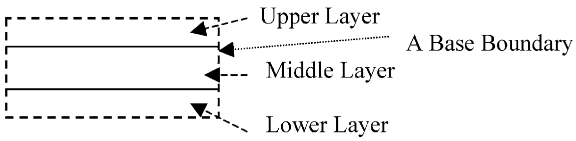

Fig. 7. Layer (zone) boundaries.

2.2. Determination of skew angle

Almost all the recognition algorithms need the text lines in the input image to be horizontal. Therefore, any skew associated with the input image needs corrections prior to recognition. Experiments show that the recognition algorithm proposed in this paper tolerates a skew of $+1^\circ$ to -1°. The accuracy of recognition deviates considerably with the increasing skew. Therefore a robust method for skew correction needs to be incorporated.

Careful observation of a line of Sinhala script shows that the boundary between the upper and the middle layers and the boundary between the middle and the lower layers (Fig. 7) possess the highest density in the horizontal direction. The horizontal projection of a sample script clearly agrees with this concept. This is due to the fact that any character in the alphabet should touch either one or both of these boundaries. Therefore, tracing the appearance of one of these boundaries in a skewed script could be used to determine the skew angle. Although any straightforward method to detect a boundary line could have been used, a more appropriate method using the linear symmetry (LS) tensor has been proposed.

The linear symmetry tensor which gives information for each pixel of the image, on how it is organised with respect to the orientation within a local neighbourhood, could effectively be used to determine the orientation of the script. In general, the orientation angle of the resultant vector of all the vectors representing the LS for each pixel of the image would provide a near approximation to the skew angle.

In order to improve the accuracy, the interference to the final result from the following components should be eliminated:

(i) edges of the image,
(ii) background of the image, which consists of pixels having random orientations of low confidence,
(iii) other pixels (within the text area) having orientations of low confidence.

The results obtained for the LS tensor derived in Section 3.3.2 yield the skew angle within $+1^\circ$ to -1° accuracy, which is well within the required accuracy for the recognition algorithm (see Fig. 8).

Fig. 8. Stages in de-skewing.

2.3. Recognition procedure

2.3.1. Testing character set

The recognition process is based on the examination of the correlation of characters in the script of image with each character of the character set through a filtering operation. The testing character set which consists of all the characters including the composite modified characters and the horizontal modifier symbols, is built by extracting characters from an LS tensor. Each character in the testing character set is filtered (one character at a time) through the LS tensor of the script in order to identify its occurrences in the entire script.

The plot of correlation at each pixel (Fig. 9) shows that, each occurrence of the character being tested gives a strong correlation. A suitable threshold that separates the required character from the rest of the characters in the script, is then determined. This procedure is conducted for each and every character of the character set. During this process, it has been observed that a total number of 35 characters amounting to 60% of the character set separates from all the other characters with a clear threshold (Fig. 9(a)) while the balance 40% confuse with one or more characters with similar shapes (Fig. 9(b)). Eight such confusing groups have been identified.

Once all the different confusing groups are identified, another level of filtering is carried out to separate each character within the confusing group. The secondary level of filtering is performed to examine the correlation of a distinct segment from one character with all the members in the group (Fig. 10). A suitable (secondary) threshold that separates each character from the rest is then determined. A further level of filtering is carried out if the confusion still occurs.

The structure of the testing character set is as follows.

Character Identifier
LS Tensor of character
Primary Threshold
Flag to indicate confusing status
Secondary Threshold (*for confusing characters*)
Tertiary Threshold (*for confusing characters*)
ASCII Value of character
Estimated Frequency
Type of Character (*determined according to the modifiable status*)

2.3.2. Recognition

The image is initially pre-processed to remove the background noise. The image is then scaled (if necessary) to

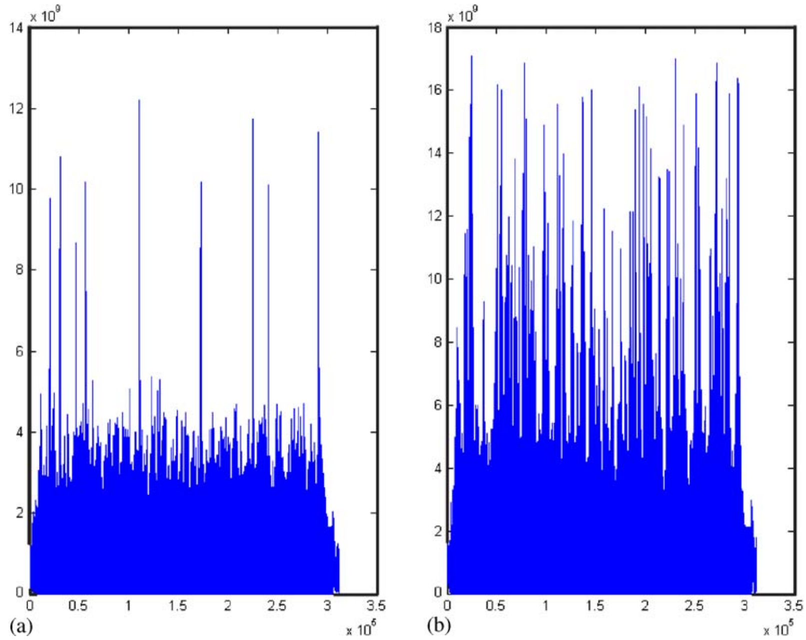

Fig. 9. Correlation of a character with the script: (a) left—a character with a unique shape; (b) right—a character confusing with similar shapes.

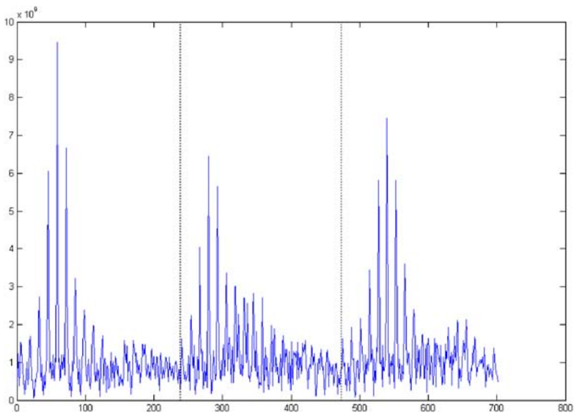

Fig. 10. Correlation of a distinct segment from one character with two similar characters.

match the average height of a character to that of the testing character set.

Recognition of the script is performed in the following two stages:

(i) Direct recognition of horizontally present characters (basic characters, composite modified characters e.g. Fig. 2, modifier symbols).
(ii) Modifiers (if) present vertically on top or at the bottom of a recognised character.

Stage (i): Filtering of the LS tensor of each character of the testing character set with the LS tensor of the script is performed. In each filtering cycle, all the occurrences of the character which is being tested, are identified. If the testing character is a confusing one, the secondary level of filtering is carried out in order to determine the acceptance or rejection of the identified character. A tertiary level of filtering is carried out similarly.

It has been observed that, in addition to the highest value of correlation produced usually at the centre of the character, a few more relatively high values are also produced around the neighbouring pixels. This is due to the fact that the template of the testing character nearly coincides with the neighbouring pixels around its centre giving a relatively high correlation. This will result in recognising the same character in the image more than once. Therefore, once the filtering of a particular character has been performed, non-maxima in a small neighbourhood (e.g. 3×3) are suppressed in order to eliminate the multiple acceptance of the same character (Fig. 11).

2.3.3. *Handling of different fonts*

The pre-determined (primary) threshold associated with each and every character of the recognising alphabet is effective only to recognise a script with identical or nearly identical font. Although the size variation of the same font could be dealt with by resizing the original image without need to change the threshold, it is necessary to investigate a method to adjust the threshold dynamically in order to recognise a

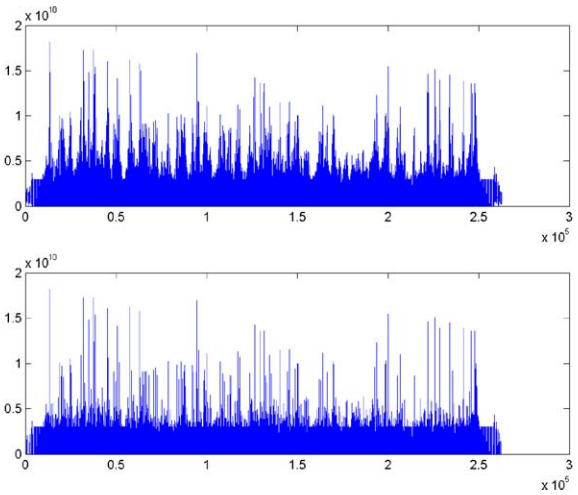

Fig. 11. Suppression of non-maximum (before and after).

script with different fonts. Therefore, the following method is proposed to generate a dynamic threshold to recognise a script with a different font. The frequency of each and every character of the character set is estimated using statistical methods. This is stored as a percentage value in a separate field in the recognising character set. The total number of characters in the input image of script is estimated using the following procedure:

(i) Separate all the lines and calculate the average line height of the script by horizontal projection.
(ii) Separate all the words in the script by applying vertical projection for each line.
(iii) Estimate the average character width by using the height/width ratio.
(iv) Use the average character width to estimate the number of characters in each word hence estimate the total number of characters in the script.

The expected number of each character (of the alphabet) in the given script could then be determined. (However, reliable values for only a very few (e.g. four), more frequent characters are required for this process.) By taking the primary threshold associated with the particular character in the recognising character set as the basis value, this threshold could now be adjusted (by applying a stepwise increasing/decreasing routine) until the expected number of characters are recognised. The final value of the threshold is the exact threshold that recognises all the occurrences of the relevant character in the given script.

A summary of the algorithm used to determine the primary threshold is given below:

(i) Determine the estimated number of characters in the given script of image.
(ii) Use the frequency of the basis character to determine the number of expected occurrences in the given script of image.
(iii) Execute the recognising algorithm to determine the number of acceptances of the basis character in the given script of image (using the standard threshold as the preliminary value).
(iv) Compare the number of acceptances with the expected value and perform the stepwise increment/decrement of the preliminary threshold value until the expected number of characters (or an approximation) is achieved.

However, the dynamic threshold calculated for a selected character of the alphabet could be used as the basis value to determine the thresholds for a subset of the alphabet with a larger number of members.

The dynamic threshold (t_x) for any member character (X) of the subset, is now given by

$$t_x = \frac{t_a}{T_a} \times T_x,$$

where T_a and t_a are standard and dynamic thresholds of basis character, respectively, and T_x is the standard threshold of (any) character X.

Stage (ii): For each recognised modifiable character (see Fig. 12), testing for the presence of vertical modifiers and identification (if present) is performed in the this stage. Using the 'type' field of a character which provides the modifiable status and the centroid of the character at which it has been recognised, for each modifiable character, its virtual boundaries without modifiers (i.e. in its basic form) are determined. The upper and lower boundaries are then extended to meet the upper and lower boundaries of the (virtual) line (Fig. 13). The extended rectangular area in the respective LS tensor of the image (above and below the basic character position) is then examined for the presence of possible edges. As the presence of an edge indicates the presence of a modifier, the LS tensor of the same area is filtered with LS tensors of appropriate modifier symbols in order to recognise the exact modifier symbol.

The recognition algorithm is as follows:

Input Image of Script
Input File of Character Set (*Section 3.2.1*)
Pre-process Image to remove background noise
Perform horizontal-projection to extract Line-Data and adjust image size
Construct LS-tensor (I_{20})
Read First Character (*from the file of character set*)
While not-end-of-character-set do
  Determine the Primary Threshold (*for the current character*)
  Filter Current Character with the LS Tensor of Image $*/$*Primary Filtering* $/*$
  Suppress Non-Maxima in the resulting Image

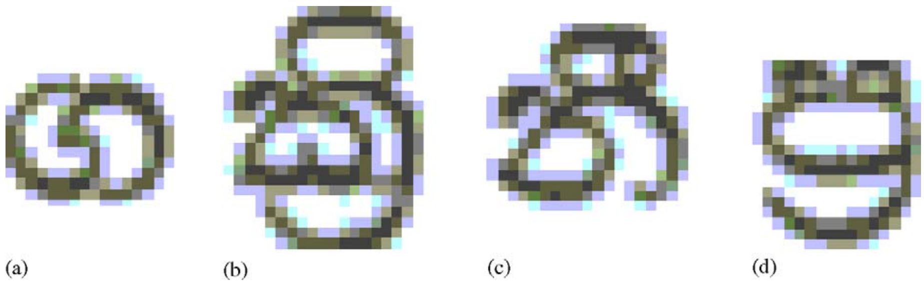

Fig. 12. Classification of recognised basic characters: (a) no modification, (b) modifiers on both top and bottom, (c) modifiers on top only, (d) modifier at bottom only.

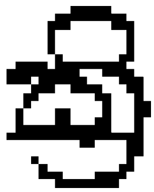

Fig. 13. Recognition of vertical modifiers.

```
            While not-end-of-image do */for each pixel of the
              image/*
              Identify any occurrence of the character */
                correlation above primary threshold/*
              Get (isolate) the identified occurrence from the
                image
              If a confusing-character
              Determine secondary threshold
              Perform secondary-filtering /* and tertiary-
                filtering if necessary*/
            End-If */ a confusing character/*
            Store image-co-ordinates (i.e. Row Number,
              Column Number) of the occurrence
          End-While */ not-end-of-image/*
          Add all recognised occurrences of current character
            to Output Character Array
          Read (next) Character
End-While */not-end-of-character-set/*
For each recognised character in the Output Character
  Array
      If Character is modifiable and a modifier is present
          Recognise Modifier
            Store Modifier co-ordinates in Modifier Array
End-For */Output Character Array /*
Merge Modifier Array with Output Character Array
Sort Output Character Array on Column No .within the
  Row No.
Write Output Character Array to a Text File
```

Since a character is identified directly within the image of the script, the need to segment individual characters prior to recognition does not arise. Symbols such as comma, full stop, question mark are also recognised in the same manner.

3. Experimental results

3.1. Skew correction

The proposed method to determine the skew angle using linear symmetry was compared with the straightforward projection method and a method used for the image in the frequency domain [7]. In most of the cases, performance of the proposed method was slightly better than the other two methods (Table 1).

3.2. Recognition

Experiments were conducted using a variety of script images for the proposed method. Since no extensive research has been done for the recognition of the Sinhala script, in

Table 1
Experimental results—determination of skew angle

Skew angle set in the original image	Experimental values		
	Using LS image	Using projection method	Using frequency image
17 (R)	17.60	17.33	18.02
38 (R)	37.85	39.42	38.20
57 (R)	57.27	59.05	58.67
7 (R)	7.27	7.20	8.04
4 (R)	3.74	6.50	4.72
85 (R)	84.41	NA	83.70
27 (L)	25.84	26.40	26.81
63 (L)	62.94	NA	60.00
16 (L)	15.07	17.25	14.33
3 (L)	3.00	1.06	NA
6 (L)	6.05	6.22	8.83
17 (L)	17.81	16.20	16.70

(R)—right skew, (L)—left skew, NA—no acceptable results.

Table 2
Experimental results—recognition of script

Font type	Recognition rate	
	Proposed method	Bench mark method
Same as recognising character set	84–93%	55–62%
Different from recognising character set	75–88%	

addition to these experiments, a segmentation based method presented in Ref. [3] to recognise Brahmi descended Bengali characters, was also tested as a bench mark for the same images of Sinhala script in order to compare the performance of the proposed method.

The experimental results as shown in Table 2, are classified into two categories namely, the recognition of the font which is identical to that of the recognising character set and the recognition of different fonts.

Each image carried 600–1200 characters and the quality of images varied from more noise to less noise and some of the images were captured from the newspapers of low quality prints.

4. Conclusions and future work

The results of the experiments establish an effective foundation to use linear symmetry as the principal feature for the recognition of printed Sinhala characters. Since the Sinhala language and the script have been evolved more independently in the island of Sri Lanka, without a close connection to other South Asian languages in the Asian sub-continent, some of its characteristic features need to be handled distinctly. The experiments conducted for the alternative method indicate that the errors in segmentation contribute largely for the errors in recognition. Since the segmentation of individual characters is not required in the proposed method, more attention could be paid to minimise the errors in the recognition process. Also, it is an advantage that the linear symmetry itself is used to separate confusing characters because no additional implementations and computations needed.

In the future work, it has been planned to incorporate, the characteristic features of the Sinhala script (e.g. a vowel is used only as the first character of a word), statistical methods, the use of contextual information such as an on-line dictionary. The experiments being conducted to use the ANNs to improve accuracy in the classification of characters within confusing groups show a further improvement.

Acknowledgements

Support of SIDA (Swedish International Agency for Development Co-operation) is gratefully acknowledged.

References

[1] A.K. Dutta, A generalised formal approach for description and analysis for major Indian scripts, J. IETE 30 (1984).
[2] S.M.S. Rajasekaran, B.L. Dekshatulu, Recognition of printed Telugu characters, Comput. Graphics Image Process. 6 (1977).
[3] A. Dutta, S. Chaudhury, Bengali Alpha-numeric character recognition using curvature features, Pattern Recognition 26(12) 1757–1770.
[4] Shamik Sural, P.K. Das, An MLP using Hough transform based fuzzy feature extraction for Bengali script recognition, Pattern Recognition Lett. 20 (1999) 771–782.
[5] J. Bigun, G.H. Granlund, Optimal orientation detection of linear symmetry, ICCV'87, London 1987, IEEE Computer Society Press, Los Alamitos, 1987, pp. 433–438.
[6] B. Jähne, Digital Image Processing—Concepts, Algorithms and Scientific Applications, Springer, Berlin, 1997.
[7] H.L. Premaratne, J. Bigun, Recognition of printed Sinhala script using linear symmetry, Fifth Asian Conference on Computer Vision, January, 2002, pp. 813–317.

About the Author—HEMAKUMAR LALITH PREMARATNE received the B.Sc. from the University of Peradeniya, Sri Lanka, PG Diploma in Computer Technology and the M.Sc. in Computer Science from the University of Colombo, Sri Lanka and the Licentiate in Engineering Degree from the Chalmers University of Technology, Sweden. Currently, he is studying for the Ph.D. degree at the Halmstad University, Sweden. He is a Lecturer at the University of Colombo School of Computing, Sri Lanka.

About the Author—JOSEF BIGUN obtained his MS and Ph.D. degrees from Linkoeping University in Sweden in 1983 and 1988, respectively. In 1988, he joined the Swiss Federal Institute of Technology in Lausanne as Adjoint Scientifique where he worked until 1998. He was elected Professor to the Signal Analysis Chair, his current position, at Halmstad University and Chalmers University of Technology in Sweden in 1998. He has co-chaired several international conferences. He has been contributing as a referee or as an editorial board member of international journals including Pattern Recognition Letters and IEEE Image Processing. He served in the executive committees of several scientific associations including the International Association for Pattern Recognition, IAPR. His scientific interests include a broad field in computer vision and pattern recognition including biometric signal analysis, texture analysis, motion analysis, 3-D modelling and understanding of the biological recognition mechanisms of audio-visual signals. He has been elected to the grade of Fellow of IAPR and IEEE.