

OPTICAL CHARACTER RECOGNITION FOR SINHALA TEXT USING FEATURE ANALYSIS

G. K. Watugala and W. M. P. Kumara
Dept. of Mechanical Engineering, University of Moratuwa

OPTICAL CHARACTER RECOGNITION FOR SINHALA TEXT USING FEATURE ANALYSIS

G. K. Watugala and W. M. P. Kumara
Dept. of Mechanical Engineering, University of Moratuwa

ABSTRACT

Optical Character Recognition (OCR) is now a reality for documents printed in English. In the present study, the groundwork for the recognition of Sinhala characters is done. Matrix Matching and Feature Analysis are the two commonly used methods for the recognition of English letters. In this study the Feature Analysis method is investigated to recognize Sinhala characters.

Matrix matching method is found to be suitable for recognizing documents containing text with known font and typeface. It should also be used to identify and extract the modifiers used on top, down or after the character. This helps in the identification of the base character using feature analysis.

Several features of Sinhala characters can be extracted by running simple programs on the pixel array of the character. These features include aspect ratio, inscribing octagon, and number of pixel curves crossed when the character is sliced at different angles. By running these programs on Sinhala characters one can prepare a set of values of these features for standard characters. Afterwards the features of an unknown character can be compared with the standard data for recognition.

Programming for Sinhala Character Recognition is done using MathCAD, an application package for complex mathematical calculations. Since the algorithms are written in pseudocode it is easy to convert these algorithms to a C++ program.

INTRODUCTION

Researchers have been working on the use of Optical Character Recognition (OCR) for scanned documents written even for languages such as Chinese and Korean which have complex alphabets (1, 2). OCR is now a reality for documents printed in English and it has a high reliability.

Sinhala text is now widely used in electronic documents in Sri Lanka. This is because of the widespread use of computers in education, government, and in industry. Because of the large number of characters in the Sinhala alphabet, the keyboard configurations for entering Sinhala are very complex. In addition, there is no standard keyboard configuration, and hence the keyboard configuration varies from one office to the other.

Thus it is very useful if we can enter Sinhala text into the computer by some kind of Optical Character Recognition (OCR) software operating on a computer scan of a handwritten or printed page. After entering the text this way, the user can fix any missing or misidentified characters manually. Later on, the user can format the document as needed.

The operation of OCR is as follows: The program first checks the overall layout of the text in the page. It separates rows of text and then isolates each and every character (or parts of it such as ො) that makes up the row in each row. This task is needed even in OCR of English text and therefore it is not dealt with here. After the characters are isolated they are subjected to matrix matching and various feature recognition rules.

In the matrix matching method, the scanned document is searched to see if it has scanned images of standard Sinhala letters already in storage. This is a straightforward method but has less chance of success if the document has Sinhala characters in different fonts or sizes.

On the other hand, feature analysis has wider application because it is less prone to variations in printing or fonts or style in the case of handwriting. The rules for identifying Sinhala characters are based on the special features (such as rounded shape and straight-line portions) of characters. Although some characters may have very unique features using which they can be identified, the computer codes required for checking those features may be very complex making them worthless in practice. Thus this paper deals with features which do not require a large effort to extract.

RECOGNITION OF CHARACTERS USING FEATURE ANALYSIS

Feature Group 1: Features that Relate to Outside Limits of the Character

Several features of the character can be obtained by first checking the outside limits of the character. These are easy to obtain by running a simple MathCAD program on the scanned image of the character as input data. MathCAD is a powerful computer application package for complex calculations. It allows reading of bitmap images and has programming facility too (3). MathCAD programs were written to obtain the following features in this study.

1. Inverse Aspect Ratio (Height/Width ratio) of the character
2. Size and shape of the octagon that can enclose the scanned image of the character
3. Pixels of the image that lie on the boundaries. e.g. The leftmost top pixel, leftmost bottom pixel of the character

Inverse Aspect ratio

A computer code was written to analyze the scanned images of characters and the results are given in Table 1 for a list of selected cases. The MathCAD program to

obtain the leftmost top pixel of the character is given in the Appendix. Here the pixels with ink (black colour) are assumed to be denoted by 0 and the blanks (white colour) are denoted by 1. The pixel array of the character is transferred to a 2-dimensional array in MathCAD.

Table 1: Inverse Aspect Ratios of Some Selected Sinhala Characters

Character	ඳඳඳ	ගගග	ආආආ
Minimum	1.4	0.64	1.58
Average	1.5	0.70	1.78
Maximum	1.6	0.77	1.92

Size and Shape information of the octagon that can enclose the character

Similarly, the octagon that can enclose the character can be found and a table can be given to list the normalized values of chopping distances.

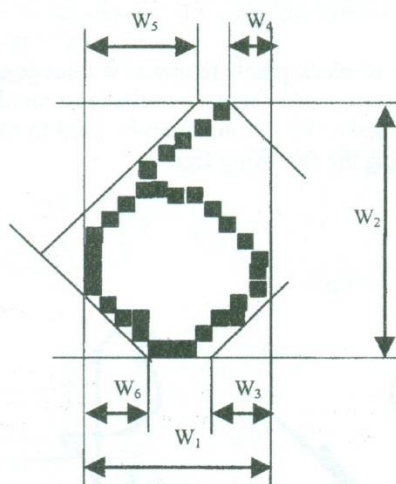


Figure 1: Inscribing Octagon of a Character

The distances from the corners to the character are given by measuring the intercept of various cropping lines with the horizontal line as shown in W_3, W_4, \dots, W_6 (Fig. 1). These distances are divided by W_1 and the results are given as percentages in Table 2.

Programs were written to find the NE, NW, SW, and SE boundaries of the character, and one of the programs is given in the Appendix as an example. By running these programs, the values given in Table 2 have been obtained.

Table 2: Size and Shape of the Inscribing Octagon of the Image of Character

Character	NE ($100W_4/W_1$)	NW ($100W_5/W_1$)	SW ($100W_6/W_1$)	SE ($100W_3/W_1$)
6	43	57	28	43
4	33	42	42	33
9	36	33	43	21
0	25	25	38	25

Feature Group 2: Features that Depend on the Arrangement of Pixel Curves when viewed at Different Directions

The Direction in which the Character has the Longest Array of Black Pixels

Using a simple program, the number of black pixels in any of 8 directions mentioned above can be tabulated. The path that gives the largest and the least number of pixels in these 8 directions has been noted down. This feature can be used to identify many characters. This can be illustrated using the following figure.

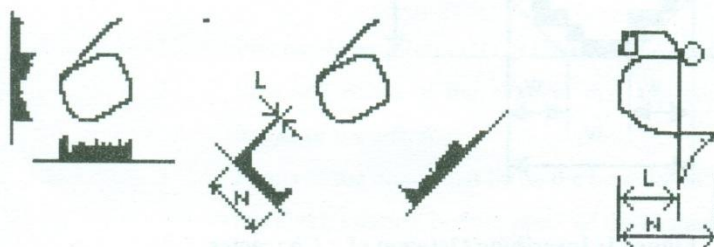


Figure 2. Orientation of the Longest Line of Pixels

In character "rayanna", the longest straight array of pixels is found along a line inclined to horizontal by 45° . Because of the circular portion of the character, there is no such array of pixels in other directions. Hence this is a good test to identify 0.

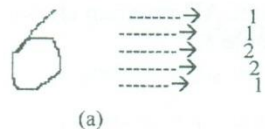
Table 3: Directions of Lines with the Largest Number of Black Pixels for Selected Characters

Shape of Character	Along horizontal Lines	Along vertical Lines	Along NE to SW Lines	Along NW to SE Lines
6	Not available as a checking rule	Not available as a checking rule	L/N=1/14 pixel Ratio	Not available as a checking rule
5	Not to be used as a checking rule	L/N=16/27 pixel ratio	Can not be used as a rule for checking	Can not be used as a rule for checking

Feature Group 3: Number of Pixel Curves Crossed when a Character is sliced along Different Directions

The shape of a character can vary from one font to another font. In the case of handwritten letters the shape varies from person to person. However, if the character is sliced at uniform intervals along various directions, the number of pixel curves encountered by the cutting edge may be the same. This feature is independent of the shape of the character. For example one may write a curvy letter, an another may write a letter using straight lines. Although the other features mentioned in this paper may be different for these two characters, feature 3 may be the same. This may be illustrated using Figure 2.

Character Number of pixel curves encountered by horizontal Lines at different levels



(a)



(b)

Figure 3. Slicing codes of a Character

Thus for horizontal slicing of the character in Fig. 3(a) we get the code 012210. Depending on the shape of the character it is also possible to get the code 0123220 as shown in Fig. 3(b). The middle "22" should be condensed to a single "2" because the cutting lines sees the same pixel curves. Table 4 lists the values of this code for several characters.

Table 4: Slicing Codes for a set of Selected Characters

Character	Horizontal Slicing	Vertical slicing	NE to SW slicing	NW to SE slicing
ඳ	012210 or 0123210	0123210,012320	01210, 0123210	0123210
උ	012321210	0123210	0123210	0123210
ඌ	012320	0121210	0123210	0123210

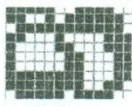

The MathCAD program to obtain slicing codes of a character is given in the Appendix. By running a more sophisticated program it is even possible to obtain a code which gives more details on the ink strokes of the characters.

Feature Group 4: Features Relating to Blank Pixels in the Character

The location and the size of blank pixels of the character are also good decisive factors in the recognition of Sinhala characters. Some of the blank (white) pixels are fully surrounded by ink (black) pixels. So an OCR program should differentiate between inside blank regions and outside regions. Once regions are identified, their areawise size and the centre of gravity should give us some means of identifying the character.

Number of closed regions, area of closed regions, and centre of gravity of closed regions has been calculated for many characters using the MathCAD program shown in the Appendix. The results for two characters are given in Table 5.

Table 5. Analysis of Location and Size of Closed Regions inside the Character





Character	Closed Region Number	Area as a percentage of whole number of pixels	Distances to CG as percentage of maximum
	1	2	(27,27)
	2	12	(61,30)
	3	13	(66,73)
	1	1	(35,27)
	2	6	(63,76)
	3	10	(63,44)

EXPERIMENTAL RESULTS

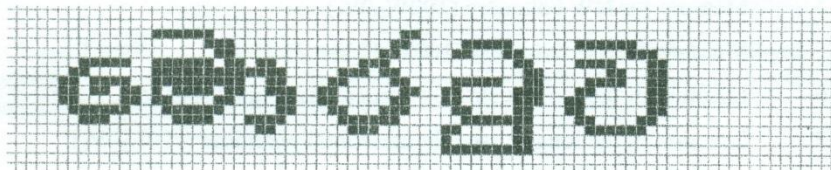
As a test, the following features of a collection of ϕ were collected and the results are given in Table 6.

1. Inverse aspect ratio ($100W_2/W_1$)
2. Measure for SE boundary ($100W_3/W_1$)
3. Measure for NE boundary ($100W_4/W_1$)
4. Measure for NW boundary ($100W_5/W_1$)
5. Measure for SW boundary ($100W_6/W_1$)
6. Location of the longest straight vertical pixel line ($100L_1/W_1$)
7. Location of the longest straight horizontal pixel line ($100L_2/W_2$)
8. Location of the longest straight SW→NE pixel line ($100L_3$ /SW to NE width)
9. Location of the longest straight SE→NW pixel line ($100L_4$ /SE to NW width)
10. Horizontal Slicing code
11. Vertical Slicing code
12. Code for slicing from SW to NE
13. Code for slicing from SE to NW
14. Number of enclosed blank regions

Table 6: Results of Feature extracted from a sample of Ó

Feature					Minimum Value	Average Value	Max. Value
1	162	150	144	138	138	148	162
2	19	25	6	33	6	20	33
3	0	0	0	27	0	7	27
4	63	69	56	61	56	62	69
5	13	6	6	5	5	7	13
6*	69	75	6	5	5	39	69
7	46	87	91	36	36	65	91
8	22	21	13	8	8	16	22
9*	85	85	50	14	14	59	85
10	121	121	121	121	-	-	-
11	12321	12321	1232	1232	-	-	-
12*	13421232321	242314231	131531	21202121232	-	-	-
13*	12123212	1231	13214231	10124321212 021	-	-	-
14	1	1	1	1	1	1	1

Using data gathered a sample document shown in Figure 4 were analyzed to see if we can identify Ó in it.



Feature #	2, 3	2,	2, 3, 4	2,	2, 3
.that	6,7,8	6, 7,	6, 7,	6, 7,	6, 7
match			10,11,		
with those		14	14	14	
of Ó in					
Table 6					

Figure 4: Sample text used for testing the identification of Ó.

Table 7: Features extracted from entities in the sample text in Figure 4

Entity →	අ	ඔ	ඌ	ඹ	ඵ
Feature ↓					
1	88	79	111	122	100
2	25	14	33	11	22
3	13	36	0	33	22
4	25	0	66	22	11
5	25	21	33	0	22
6	63	43	80	90	90
7	43	45	91	100	10
8	11	67	27	77	53
9	-	27	44	11	56
10	12321	13124321	121	1212121	121
11	12321	23121	1232	24321	123431
12	12321	1231321	121	12342321	1232321
13	123232	1234121	121	123232121	12321
14	0	1	1	1	0

It is demonstrated from the test that Sinhala characters can be identified from feature analysis.

CONCLUSIONS

Many features of Sinhala characters can be extracted by analyzing the bitmap monochromatic image of the character. The algorithms for extraction of several features were given in this paper as groundwork for a complete set of feature analysis programs needed for the successful implementation of OCR.

The algorithms were implemented using MathCAD application package. The results obtained with these programs indicate that the algorithms are applicable in practice.

REFERENCES

1. T. S. Agui and H. Nagahashi, A description method of Handprinted Chinese Characters, IEEE Transactions on Pattern Recognition and Machine Intelligence January 1974, Vol. 1, p. 20-25
2. S. Kreada, A Method of Recognition and Representation of Korean Characters by Tree Grammers IEEE Transactions on Pattern Analysis and Machine Intelligence, July 1979, No. 3, p 245-251
3. MathCAD reference manual (version 6.0), Publisher: MathSoft Inc. UK. 1994.

APPENDIX

In the following algorithms it is assumed that the character is written in black. Black pixels are denoted by 0 and white pixels are denoted by 1. The character is stored in a two dimensional array and the programming is done on this array to recognize the character

Algorithm for Detecting the Topmost Left Pixel in the Character

```
WestmostTopA, key) -
    r ← rows(A)
    c ← cols(A)
    i ← 1
    j ← 1
    while (j ≤ c) · (Ai,j ≠ key)
        i ← i + 1 if i < r
        otherwise
            i ← 1
            j ← j + 1
    ( i )
    ( j ) if j ≤ c
    ( -1 ) otherwise
    ( -1 )
```

Algorithm to find the NW boundary of the Character

```

NW(A) : r ← rows(A)
        c ← cols(A)
        NW ← r + c - 2
        for i ∈ 1, 2..r
            for j ∈ 1, 2..c
                if (Ai,j = 0)
                    z ← i + j - 2
                    NW ← z if (z < NW)
        NW ← floor  $\left[ \frac{NW \cdot 100}{(c - 1)} \right]$ 

```

Algorithm for Obtaining the Slicecode of a Character

```

Slicecode(A) : r ← rows(A)
               c ← cols(A)
               Slicecode ← 0
               count ← 0
               for i ∈ 1, 2..r
                   Oldcount ← count
                   count ← 0
                   for j ∈ 1, 2..c
                       count ← count + 1 if (Ai,j = 0) · (j = 1)
                       count ← count + 1 if (Ai,j = 0) · (j = c)
                       count ← count + 1 if (j ≠ c) · (Ai,j ≠ Ai,j+1)
                   count ←  $\frac{\text{count}}{2}$ 
                   Slicecode ← 10 · Slicecode + Oldcount if (Oldcount = 0) · (count ≠ 0)
                   Slicecode ← 10 · Slicecode + count if (Oldcount ≠ count) · (count ≠ 0)
               Slicecode ← Slicecode

```

Algorithm to find Area and Centre of Gravity of Closed Regions

```

CGcalculate(A) := A ← CRNumbering(A)
N ← A1,1
r ← rows(A)
c ← cols(A)
for n ∈ 1, 2..N
    if (N ≥ 1)
        j1 ← (n - 1) · 3 + 1
        j2 ← (n - 1) · 3 + 2
        j3 ← (n - 1) · 3 + 3
        sumw ← 0
        sumx ← 0
        sumy ← 0
        for i ∈ 2, 3..r
            for j ∈ 1, 2..c
                if Ai,j = n + 1
                    sumw ← sumw + 1
                    sumx ← sumx + i
                    sumy ← sumy + j
        xbar ←  $\frac{(\text{sumx}) \cdot \left(\frac{1000}{r}\right)}{(\text{sumw})}$ 
        ybar ←  $\frac{(\text{sumy}) \cdot \left(\frac{1000}{c}\right)}{(\text{sumw})}$ 
        Area ←  $\left[ \frac{\text{sumw} \cdot 1000}{(r \cdot c)} \right]$ 
        A1,j1 ← floor(Area)
        A1,j2 ← floor(xbar)
        A1,j3 ← floor(ybar)

```

A