Vol. 1 No. 1 情報知識学会誌 Dec. 1990

PAPER

Automatic Devanāgarī Character Recognition Using Structure Analysis

Krishnamachari Jayanthi[†] Akihiro Suzuki[††] Hiroshi Kanai[†††] Youko Akiyama[†††]
Yoshiyuki Kawazoe[††††] Masayuki Kimura[†] Ken'iti Kido[††] Keisho Tsukamoto[†††††]

Devanāgarī characters, which are used in Sanskrit, Hindī, and other Indian languages, are the target of automatic recognition in this study. The present research is confined to recognizing 67 printed characters which occur frequently in the old Sanskrit manuscripts of the Saddharmapuṇḍarīka. The prominent features of these characters are extracted and the top-doun binary tree is used for their recognition. A special thinning algorithm is also developed for this script. The recognition rate obtaind is more than 95% for the extracted characters.

1. Introduction

Devanāgarī is the most widely used script in India. Just as Kanji is used both in the Chinese and Japanese languages, Devanāgarī is used in Sanskrit, Hindī and Marathī, the latter two languages being included among the official languages of India, and Hindī speakers forming 30% of the population. Sanskrit is a language with a history similar to that of Latin; widely used by the ancient scholars in their literary compositions, hence of great academic interest, but not in everyday use at present.

The Devanāgarī alphabet consists of 14 vowels, 33 consonants, 10 numerals and 3 special characters. Not all the characters are necessarily present in any given text; there are even characters which occur only in combination forms, and some characters may be found only three or four times in the entire text. If we include the vowel-consonant (v-c) and consonant-consonant (c-c) combinations, the number increases enormously. The v-c combinations total 462 (14 vowels X 33 consonants). The c-c combinations can be formed by adding any number of consonants in any order, but in practice, the number of c-c combinations is quite limited and may not exceed fifty for texts of ordinary complexity (number of characters = 14 vowels by 50 c-c's = 700). The c-c combination characters are formed by restoring the characteristic features of elements. A detailed explanation of the combination of these elementary fonts to form combination characters in Devanāgarī can be found in Ref[1].

Many of the original texts of Buddhist literature are written in Devanāgarī (Sanskrit). Given the widespread interest in the study of original Buddhist texts and considering that it is quite unrealistic to expect those engaged in Buddhist literature studies at present to learn an entirely new script for the purpose of their research, the task of converting texts written in

[†]Faculty of Engineering, Tohoku University Aramaki aza Aoba, Aoba-ku, Sendai 980, Japan
[††]Research Center for Applied Information Sciences, Tohoku University 2-Choume 1-1, Katahira, Aoba-ku, Sendai 980, Japan
[†††]Education Center for Information Processing, Tohoku University Kawauchi, Aoba-ku, Sendai 980, Japan
[††††]Institute for Materials Research, Tohoku University 2-Choume 1-1, Katahira, Aoba-ku, Sendai 980, Japan
[†††††]Faculty of Arts and Letters, Tohoku University Kawauchi, Aoba-ku, Sendai 980, Japan

Devanāgarī into Roman script becomes necessary. Since the Buddhist texts are voluminous, a computer system to automatically transcribe Devanāgarī into Roman script would be eagerly welcomed. Further, printed data, due to the aging of the paper, is always subject to deterioration and the cost and effort of restoration is also high. If the texts could be stored in a computer database, they could be more easily preserved and very easily maintained in terms of cross-references and so on. A beginning has been made in this direction by concentrating on a specific text.

Sinha and Mahabala[2] once tried to recognize Devanāgarī automatically according to their pattern analysis system. They chose 26 symbols and extracted structual information for these characters. They tested their method experimentally using the IBM 7044 system. However, their experiment was limited in sample size (several characters for each symbol) and could not give a quantitative recognition rate. Although the present study has been developed independently of the method devised by Sinha and Mahabala, it conceptually is an extension of their work. We have employed a more sophisticated thinning algorithm, a large set of characters, a more computer suitable feature extraction method, and an exhaustive experimental recognition test, aiming to achieve a practical level of automatic Devanāgarī recognition. Since one of the final aims of this project is to recognize automatically the old handwritten Buddhist manuscripts in Devanāgarī, structure analysis method is essential. We are also studying pattern matching method to recognize printed Devanāgarī script, and a part of the result has been published in Ref[1].

This paper is divided into four sections. The next section describes the method of data collection and the problems encountered due to the age of the text. The thinning algorithm and the modifications made to the existing algorithm in view of the specific requirements of Devanāgarī are presented in the third section. The fourth section explains the actual methodology of feature analysis. The fifth section deals with the algorithm used for recognition, namely the top-down binary tree, and the results. The last section is devoted to the conclusion.

2. Data Collection and Problems

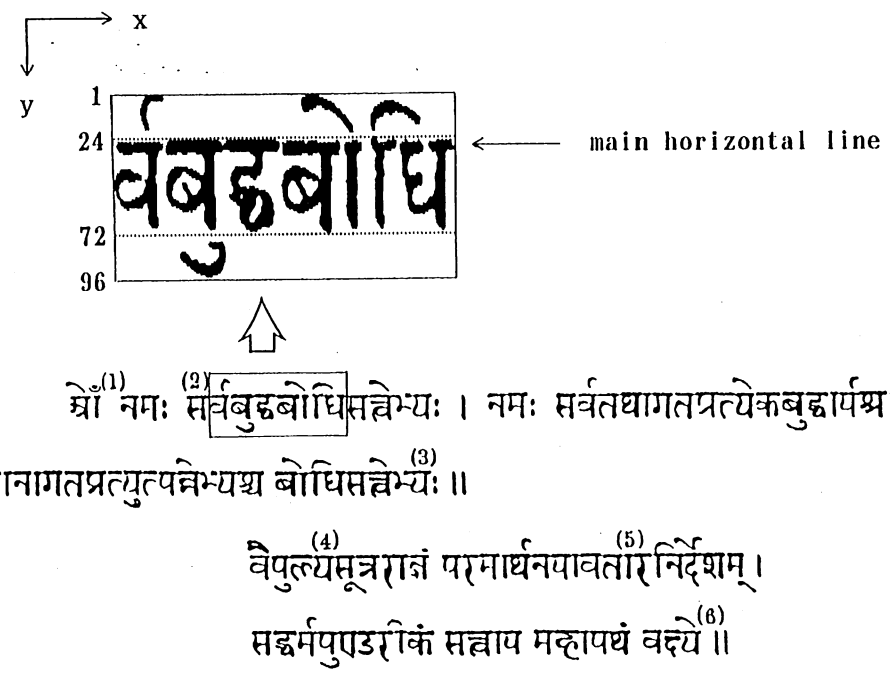
2.1 Database

The programming and database creation was done on IBM 3081-KX6. The database consists of whole lines of text and the coordinates of extracted characters for each line of text. The original text has been expanded by a factor of 2.5 and was read in sequence (line by line) using an image scanner (IBM 6392). Manual adjustments were made to make sure that the page was properly aligned. The characters stored are in the form of a 144 by 96 matrix. Though it cannnot be said that even with this high resolution the characters are being read with 100% accuracy, practical limitations such as storage and processing speed make it difficult to handle matrices larger than the one mentioned above.

The extraction has been performed by the standard histogram method. The lower part of Figure 1 shows the first page of Saddharmapuṇḍarīka[3], printed in Devanāgarī. As is shown in Fig. 1, due to the nature of the script, each line has been divided into three longitudinal sections, the first being from pixel 1 to 24 on the y axis, the second, which is the main part, from 25 to 72, and the third from 73 to 96. Although the extraction is done separately for each section, the three parts are integrated by examining and matching the coordinates before the character is analysed for recognition.

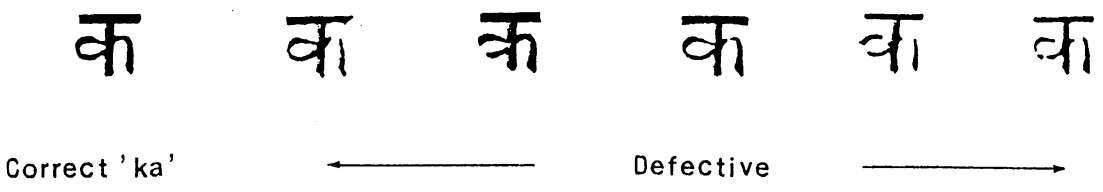
2.2 Problems in Data Collection

The Sanskrit text which was used for recognition is the Saddharmapuṇḍarīka (Lotus Sutra)[3], one of the most important Buddhist texts. This text was edited and printed in 1912 by H. Kern and B. Nanjio. However, the quality of printing



The enlarged upper part shows the present resolution and the coordinate system used.

Fig. 1 Devānagarī sample text in Saddharmapuṇḍarīka.



Except for the leftmost one, recognizable defects are observed.

Fig. 2 Example of printing defects in the text for 'ka'.

leaves much to be desired. In addition to uneven printing, it contains a lot of noise and breaks which distort the characters. Fig. 2 shows examples of these defects for the Devanāgarī character "ka" , found in Saddharmapuṇḍarīka. The problems resulting from the printing defects have been examined in detail in section 5.3 and suggestions for corrections have been made. Though the image scanner used has a resolution of 8 dots to a millimeter, as the text is copied before being fed to the image scanner, some of the fine details of the characters are lost in the final image.

2.3 Characteristics of Devanāgarī

As shown in Fig. 1, most characters have a heavy horizontal line somewhere near the top, and likewise, end a few pixels above the bottom. The region between this main horizontal line (MHL indicated in Fig. 1, which falls somewhere around 30 pixels from the top, and the lower limit, which is around pixel 72, is referred to as

the main character. There are several characters which do not have the MHL and also a few characters which extend below the usual lower limit. Generally, the separate features above the MHL or below the lower limit are vowel additions to the main character and are handled separately.

The point to be noted is that, unlike Kanji, Devanāgarī characters cannot be confined within uniform squares. Apart from extensions below the usual lower limit, even the width varies widely from the thickest to the thinnest character. There are frequent cases where two or more characters have been joined together to form combination characters and there are a few characters which are in two separate pieces. All this leads to severe problems when trying to extract separate characters from the general text. However, at the recognition stage itself, these peculiarities make it easier to identify outstandingly different characters, though extra processing is required to find character boundaries.

In the introduction, the total number of possible characters occurring in the Devanāgarī alphabet has been mentioned. Given in the next paragraph is the number of characters computed with specific reference to the text (the first 50 pages of the text were scanned for this purpose), taking the standardisation of most v-c combinations into account.

There are 35 basic characters: 6 vowels and 29 consonants. Additionally there are 4 special characters, 10 numerals, and here we include 18 frequently occurring c-c combinations for the recognition procedure. The characters considered in the present paper are listed in Fig. 3. The vowel features to be recognised are 10. Hence the total number of characters to be recognised comes to about 100, with 10 vowel features extra. The infrequently occurring c-c combinations have generally been omitted.

3. Thinning Algorithm

3.1 Holt's Thinning Algorithm

The basis for the thinning algorithm used in this study is the one developed by Holt et al.[4]. This algorithm was first applied without any changes to the characters and the results were studied. Fig. 4 shows an example of this procedure for the Devanāgarī character "a".

As mentioned earlier, two major features of Devanāgarī characters are the main horizontal line and various vertical lines. It can be seen that neither feature has been preserved by the Holt algorithm. Considering that no general algorithm would be able to take care of the special features of any particular script and that the Holt thinning algorithm was otherwise satisfactory, it was decided to modify this algorithm in order to make use of it for the special case of Devanāgarī characters. In Devanāgarī characters, MHL and vertical lines, which contain almost no information, occupy large portions of the character images and should be separated out for recognition. The changes made to the above algorithm are mentioned below.

3.2 Modifications to The Holt Thinning Algorithm

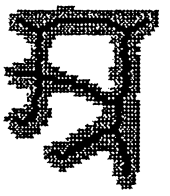
Two modifications have been made at two different stages in the algorithm.

a. The pre-processing stage : Before the actual thinning algorithm is applied, the character is examined for the presence of the main horizontal line and vertical lines. This is done by computing the histogram along with the vertical and horizontal axes and by checking for peaks. In the case of the main horizontal line, the position is also checked as this line occurs (if at all) between the same coordinates approximately, due to prior positioning during image scanner reading. For a vertical line, two checks are performed : (1) the histogram value should be nearly equal (up to five pixels less, to be precise) to the height of the main character, and (2) the width of the peak should be larger than 2 pixels. If the ver-

Fig. 3 67 Basic Devanāgarī characters for recognition with pronounciations.

tical lines are located, they are thinned to their center positions and these positions are stored in an array used to store the details of the character (character array). For every character, only one main horizontal line and up to three vertical lines are allowed for; this is adequate, as no valid character possesses more lines.

b. Modifications in the algorithm: During the thinning stage, the positions already stored in the character array are used to skip the thinning checks for these coordinates in the character, i.e., for the thinned main horizontal line and vertical lines, thus these are maintained as they are. Fig. 5 shows the modified procedure; the left half of the figure shows the intermediate stage where only the MHL and a vertical line have been extracted; the right half shows the last stage after the application of the Holt thinning algorithm to the rest of the character image, except for the extracted MHL and VL.



The thinned pattern is given by the black solid line and the original pattern is shown by the fuzzy part.

Fig. 4 Result of the unmodified Holt algorithm for 'a'.

Intermediate stage

Final stage

Main horizontal line & vertical line thinned.

The left part of the figure shows the intermediate stage of the thinning procedure where MHL and VL are extracted. The rest of the character pattern (left bottom part) is thinned using the Holt algorithm to produce the final result indicated in the right part of the figure.

Fig. 5 Result of the modified thinning algorithm.

3.3 Defects in The Modified Thinning Algorithm

Since the modifications concern two areas in the character where MHL and vertical lines are expected, the defects can similarly be categorised as:

a. Failure to detect the main horizontal line

b. Failure to detect the vertical lines

In the course of testing, a balance had to be struck between detecting pseudo MHL and pseudo vertical lines on the one hand, and not detecting some of the thinner lines on the other hand. As the characters with MHL and vertical lines are larger in number than those without, detecting pseudo MHL's and vertical lines would have led to this class getting even larger and would have required too complicated checks at a later stage to reject those characters not really belonging to this class. It was therefore decided to include these characters instead in the smaller category, i.e. in the class without MHL and vertical lines where checking and rejection are simpler.

The histogram for the MHL shown in Fig. 6 shows that the peaks in the case of both ".s" and "0" follow similar patterns. Hence both have either to be rejected or to be recognised. For reasons mentioned earlier, both have been rejected. In the case of vertical lines, the width of the peak is the point of comparison.

Inspite of the defects of the present algorithm, the extraction of MHL and vertical lines before the application of the thinning algorithm is necessary to preserve the structural information of the important part of Devanāgarī characters. It might also be possible to extract the MHL and vertical lines after application of the Holt thinning algorithm to the whole character image. Although this method is more general than the present one, we have selected the latter in order to use explicitly the characteristic features of the Devanāgarī script and reduce the complexity of the analysis.

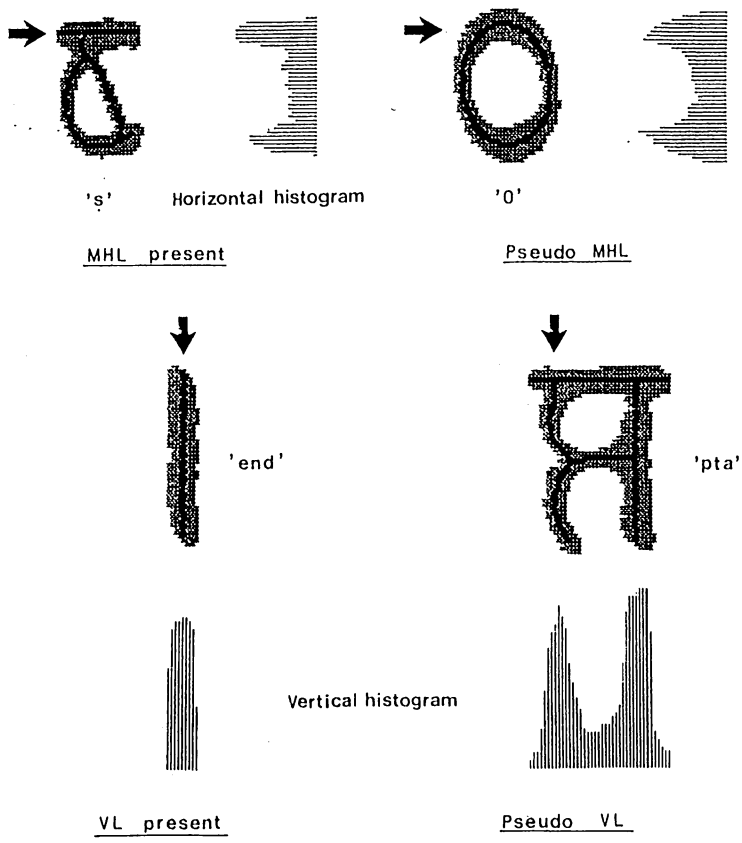


Fig. 6 Defects of the modified thinning algorithm.

4. Identification of Important Features

4.1 Method of Feature Detection

Before selecting a method for character recognition, it is necessary to take the characteristics of the script into consideration[5]. As Devanāgarī characters have widely varying sizes, unlike printed Kanji which can be confined in uniform squares, most of these methods had to be rejected as being unsuitable for Devanāgarī. The frequently used Devanāgarī characters in the text taken for study are about 100. With this small number, it is easy to locate the outstanding features and classify the characters by a binary tree based on these features.

A further reason why this method was chosen is that the method of recognising characters by their features can be extended to other fonts than the font of the text under study and subsequently to hand-written characters too. The last advantage is especially significant compared to the pattern matching method.

The features chosen for recognition fall into two categories: (1) features which are outstanding and easy to recognise in a program and (2) features which divide the characters into equal classes making the binary tree more balanced and more efficient. As much as possible, both factors were kept in mind, but the second factor was given more importance in the earlier stages of the division procedure. In the last few steps, the ease of programming was taken into consideration. It was also felt at the later stages that

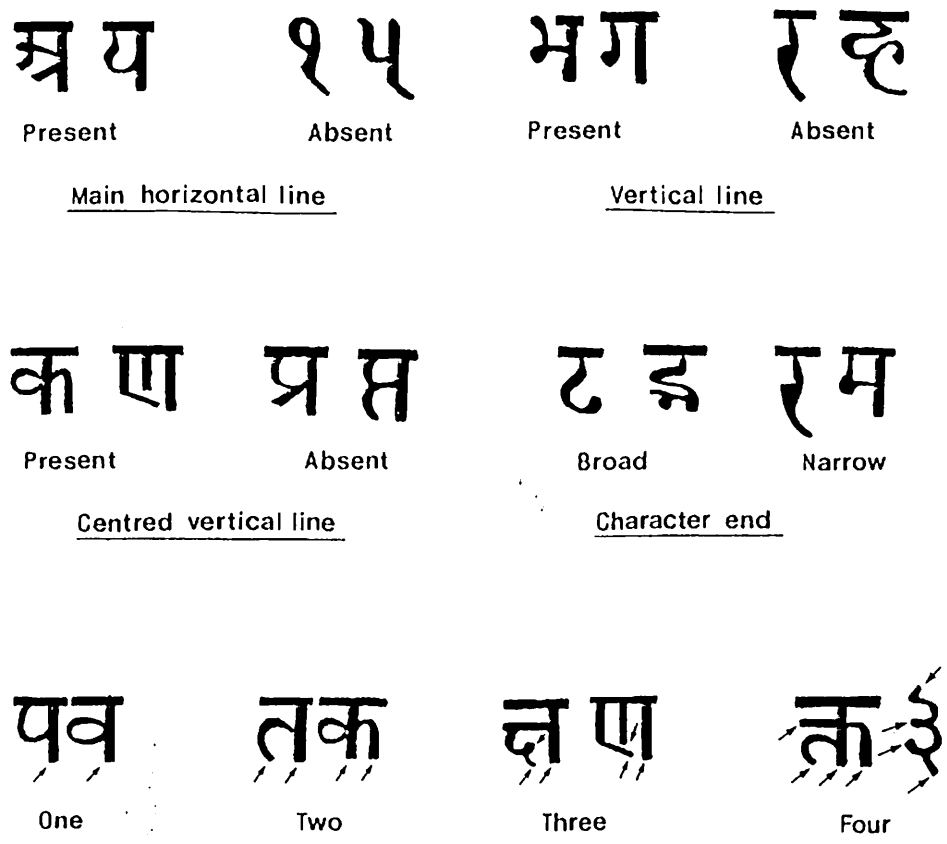


Fig. 7 Examples of some outstanding features.

it is better to use the features as they occur, instead of trying to force them into an artificial binary format. Hence at later stages in the classification tree, the structure is not strictly binary; branching into 3 or more leaves at node.

A cursory glance at the characters reveals that the most outstanding feature of Devanāgarī characters is the MHL. Although it seems at first that all the characters possess this feature, a closer inspection shows that this is not so, though the majority of characters do have the MHL. Accordingly, this was chosen as the first feature to be identified. The second feature has been the presence or absence of vertical lines, with the third (in case vertical lines are present) being whether the line is the rightmost feature in the character. The other features have been the height/width ratio of the character, whether the character is narrow or broad ended, the number of free ends it has, etc. Immense care had to be taken in the case of certain features, for example, the number and position of free ends, to take care of frequently occurring printing defects like breaks.

Examples of some of the more outstanding features are given in Fig. 7. Table. 1 gives the checklist of the features used for the recognition of each basic character (to give this

Table 1 Major features checklist for basic characters

Character	MHL	VL	CVL	MINT	PCS	END	CHAR	ENDS	FINAL LEVEL
A	Y	Y	N	2	1	N	B	4	7
I	Y	N	N	1	1	N	N	2	8
U	Y	N	N	1	1	B	B	2	6
U	Y	N	N	1	1	B	B	3	6
R	Y	Y	Y	2	1	B	B	4	5
E	Y	N	N	2	1	N	N	1	4
KA	Y	Y	Y	1	1	N	B	2	5
KHA	Y	Y	N	2	1	B	B	2	6
GA	Y	Y	N	2	2	N	N	2	7
GHA	Y	Y	N	2	1	N	B	1	7
CA	Y	Y	N	1	1	N	N	3	6
CHA	Y	N	N	1	1	N	B	1	8
JA	Y	Y	N	1	1	B/N	B	2	6
TA	Y	Y	N	1	1	B	B	2	6
DA	Y	N	N	1	1	N	N	2	8
DHA	N	Y	N	2	1	N	B	2	3
NA	Y	Y	N	1	1	N	N	2	5
TA	Y	N	N	1	1	B	B	1	6
DA	Y	N	N	1	1	B	B	2	6
DHA	Y	N	N	1	1	B	B	0	6
NA	Y	Y	Y/N	3	1/2	B	B	0-3	6
PA	Y	Y	N	2	1	N	B	1	7
PHA	Y	Y	Y	2	1	B	B	2	5
BA	Y	Y	N	1	1	N	B	1	7
BHA	N	Y	N	2	1	N	B	1	3
MA	Y	Y	N	2	1	N	B	1	7
YA	Y	Y	N	2	1	B/N	B	1	7
RA	Y	N	N	1	1	N	N	1	8
LA	Y	Y	N	1	1	B	B	3	6
VA	Y	Y	N	1	1	N	B	1	7
SA	Y	Y	N	3	2	B	B	2	6
SA	Y	Y	N	2	1	N	B	1	7
SA	Y	Y	N	2	1	B	B	2	6
HA	Y	N	N	1	1	N	B	2/3	8

Please note that the features given above are only the major features. After classifying according to the features given above, the characters are tested for the features which would be applicable in only this particular level, and hence meaningless in the larger table. For example, between 'ba' and 'va', the only difference is the stroke present or not inside the middle oval. Due to printing defects, the same character has differing features, as shown above.

MHL - Main horizontal line
VL - Vertical line
CVL - Centred vertical line
MINT - Intersections with the main horizontal line
PCS - Number of the pieces of the character
END - End of the character - broad or narrow
CHAR - Broad or narrow character
ENDS - Number of free ends of the character

The final column gives the level at which recognition took place.

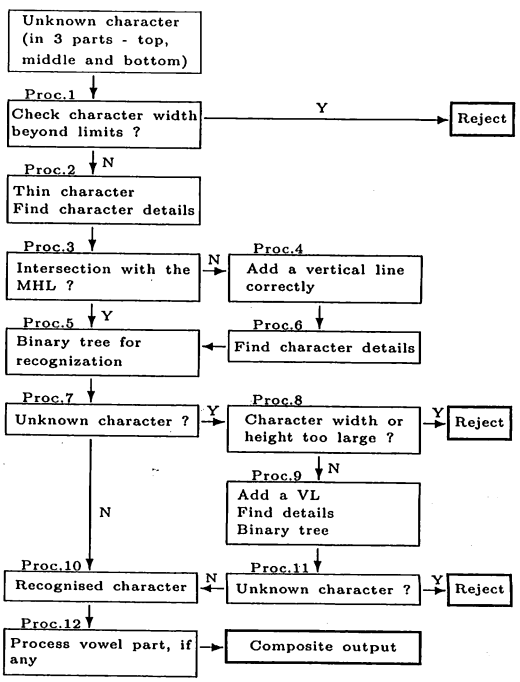
list for all the characters would have been an unwieldy exercise). Table. 2 gives the actual programming technique used to identify each of these features. The general strategy used to recognise all the characters is outlined in the next section. The characters analysed for recognition were those which resulted from the extraction program.

Table 2 Detection of major features

Feature	Method of detection
MHL	Examine the histogram between pixels 24 and 35 on the y axis and compare with other peaks to check that this is the major peak.
VL	Examine the histogram on the x axis and check for peaks of value .ge. (character height - 5) pixels & width of 3 pixels.
CVL	As the vertical line can be rightmost feature or centred (not left most), the histogram to the right of the first vertical line is checked, and if the numbers of summed pixels in the integrated area are non-zero for the main character, centred vertical line is supposed to be present.
MINT	The actual character array is scanned, for the one row just below the main horizontal line, and the number of intersections is the number of non-zero pixels in this row.
PCS	The histogram value for the main character alone is checked to find out how often a zero value followed by a non-zero value occurs.
END	The character array is scanned starting from the end of the main character till five rows above this point. If the maximum width found thus is more than half the character width, the end is supposed to be broad; else narrow.
CHAR	The maximum width of the character is checked with its total height and if the width is greater than half the height the character is broad.
ENDS	The main character array is checked in between its boundaries with a 3 X 3 window. With a nonzero pixel at the centre, if the total of the nonzero pixels among its eight neighbours is 1, or if the total is and these neighbours are consecutive, the centre pixel is supposed to be a free end. The position is also stored for further checking.

4.2 Design of The Program

The global flowchart of the program developed in the present study is given in Fig. 8. The maximum and minimum limits for the widths of all valid characters were stored in the program; these limits were computed by examining the characters occurring in the first fifty pages of the text under study. Extracted characters which were read by the program and did not satisfy the limits, were rejected even before the thinning algorithm was applied (procedure 1 in Fig. 8). The maximum limit check took care of most of the characters which could not be extracted properly, as well as some of the more complicated c-c combinations which have not been taken into account yet and have wider character widths. The minimum limit check took care of most of the isolated noise.



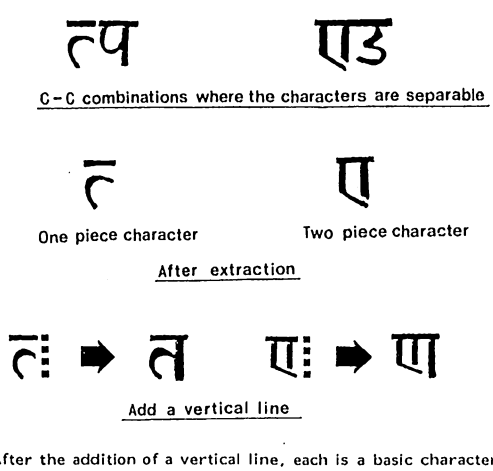
Reject procedures for unrecognizable characters are also shown.

Fig. 8 Program flowchart for Devanāgarī script recognition by structure analysis.

The next step was the pre-processing for the thinning algorithm. The character was examined for the presence of MHL and vertical lines. If these were found, they alone were thinned and the positions were noted in the character array. Subsequently, the Holt thinning algorithm was applied to the rest of the character and the thinned result obtained for recognition (procedure 2 in Fig. 8).

The various details about the character, such as the boundaries of the character, whether the character was broad or narrow ended, the free ends and their positions etc., were found and stored in the character array.

Some of the frequently occuring c-c combinations have also been taken care of in the flow of the program (procedure 4 in Fig. 8). Fig. 9 shows that two kinds of c-c combinations were taken into account in the program. The first type includes cases where the character is cut in such a way that just checking one or two details will reveal it to be a half-cut character. The second type are combinations which have to be rejected by the recognition process before they can be recognised as a half-cut character. To give further details, if the character has no intersection



Only adding a vertical line, the combination character is easily separated into two original basic characters.

Fig. 9 Handling separable c-c combinations.

with the MHL or the height of the character is too small, it is the first type. A vertical line is then added and joined to the character at the rightmost centered free end and the details of the character are found once again. In the second case, after the binary tree for the Devanāgarī character recognition (procedure 5 in Fig. 8, the first three steps of which are schematically shown in Fig. 10 and the details in Table. 3), if the height and width of the character are not too

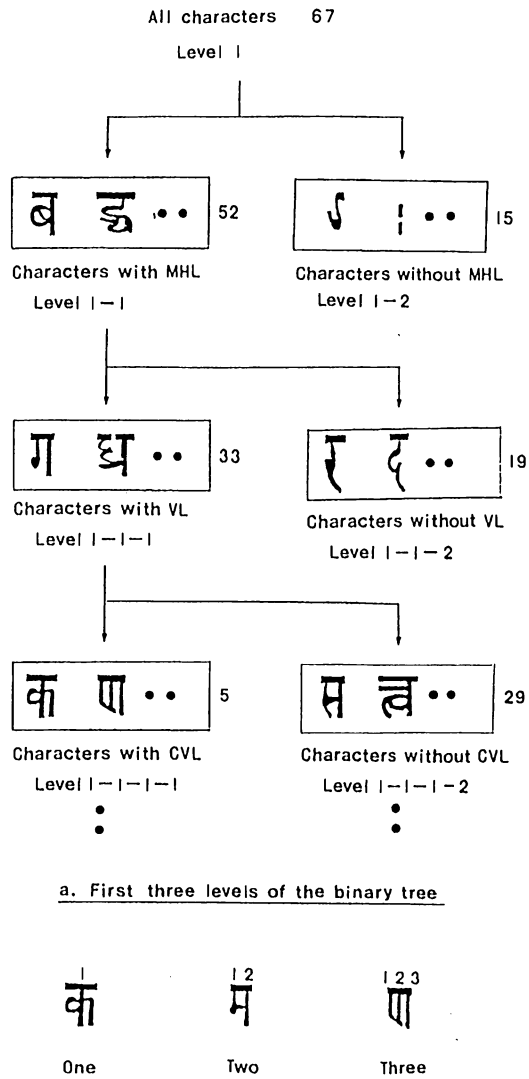


Fig. 10 Binary tree and non-binary structure.

Table 3 Basis for decision making in the tree

Level	Basis
1-1	Main horizontal line present
1-2	Main horizontal line not present
1-1-1	Vertical line present
1-1-2	Vertical line not present
1-2-1	Vertical line present
1-2-2	Vertical line not present
1-1-1-1	Centred vertical line present
1-1-1-2	Centred vertical line not present
1-1-2-1	One intersection with the main horizontal line
1-1-2-2	Two intersections 〃
1-1-1-1-1	One intersection 〃
1-1-1-1-2	Two intersections 〃
1-1-1-1-3	Three intersections 〃
1-1-1-2-1	One intersection 〃
1-1-1-2-2	Two intersections 〃
1-1-1-2-3	Three intersections 〃
1-1-2-1-1	Broad ended
1-1-2-1-2	Narrow ended
1-1-1-1-2-1	Only two free ends
1-1-1-1-2-2	Four free ends
1-1-1-2-1-1	Narrow ended
1-1-1-2-1-2	Broad ended
1-1-1-2-2-1	Disjoint character
1-1-1-2-2-2	One piece characters
1-1-2-1-1-1	Broad characters
1-1-2-1-1-2	Narrow characters
1-1-2-1-2-1	Broad characters
1-1-2-1-2-2	Narrow characters

The features given above are global and applicable to most characters. In the actual situation, due to the nature of the printing, many more detailed checks had to be performed. To take the example of the character '.NA', depending on the printing it had two vertical lines with one centred, or one only at the rightmost; three free ends, two, or none at all; either a two piece character or one. Hence to give all the details used to recognise all the characters is not possible.

large, a vertical line is added to the character (procedure 9 in Fig. 8). This, however, leads to some characters being unnecessarily processed before being rejected finally.

Note in Fig. 9 that even in this simple procedure of adding a vertical line to separate a combination character into multiple basic characters, there are two variations; namely, characters which are one piece (MHL is not counted) and characters where the vertical line is added after a gap (just adding a vertical line at the cut position, the shape of the original basic character cannot be reproduced; extracted MHL is shorter compared to the one in the basic character), in other words, two pieces. The difference between these two types is the presence and position of free ends. If there are free ends and at least one of them is at the right extreme of the character and more or less centered, the character is assumed to be a one-piece character. In all other cases, it is supposed to be a two-piece character. In the second case, the gap after which the vertical line is to be added is decided by the average width of the character.

If after performing all of these procedures the character is still not recognised, it is classified as an unrecognized character.

5. Recognition Using Binary Tree and Optimisation

5.1 Strategy for Recognition

Many of the features chosen for recognition fall into a clear binary pattern. Even the other features which do not necessarily fall into a binary pattern either can be made to do so, or else have a numerical basis for decision which is easy to program. Further, a binary tree is one of the fastest decision making processes for a computer program. For these reasons, it was decided to adopt a top-down binary tree for character recognition. The parameters for decision-making were decided on a trial and error basis in view of accommodating the maximum number of characters. Although this has led to a few errors, as explained in Section 5.3, the results are generally satisfactory (Table. 4). As there are 67 characters for recognition, to give the full binary tree is an unwieldy exercise. Accordingly, the tree has been traced up to the first three levels, with the number of characters

Table 4 Error percentages before error correction Summary of results:

Page no.	% error (total)	No.of errors	Categories of error				
			Noise	Break	Thinning	pa/ya	Others
1	1.42	5	1	2	2	0	0
2	1.83	10	3	3	1	2	1
3	1.88	8	4	0	1	1	2
4	3.36	22	6	9	1	2	4
5	3.56	17	5	7	0	1	4
6	4.44	24	5	7	7	2	3
7	7.98	48	3	20	9	3	13
8	7.34	31	6	7	9	3	6
9	11.73	48	6	23	12	1	6
10	5.68	25	3	12	2	3	5
Total	4.92	238	42	90	44	18	44

in each branch, and following the major branch only (Fig. 10a).

At later levels, the structure of the tree is not strictly binary. To give an example of a non-binary level, if we take the number of intersections with the main horizontal line, the number of branches is three most naturally (Fig. 10b). Although it would not have been difficult to force binary structure even with these kinds of features, this was not done because it was felt that it would be better to maintain simplicity in decision making rather than binary structure at any cost. All the 67 characters are classified according to this method and are the last levels in the tree. Table. 3 gives the basis for each branching in the binary tree.

By looking at the program, we can calculate the average number of levels, as well as the levels for the worst cases. The average turns out to be six, with the worst case examples at eight (see the last column in Table. 1.). Taking the average levels, the number of characters which can be recognised in a strict binary tree structure would be 64. However, as mentioned before, in some levels the branching not being binary has led to 67 characters being easily accommodated in this tree. According to the clear characteristic features extracted, the number of decisions to obtain the final candidate is only 6 on the average. This proves the effectiveness of the present binary tree method for the recognition of the Devanāgarī script.

5.2 Experimental Results

In order to evaluate the effectiveness of the method chosen, recognition experiments were conducted, taking the first ten pages of the text as sample. The total number of extracted characters in the first ten pages is 4863. This being the characters as given by the extraction program, including even those not properly extracted.

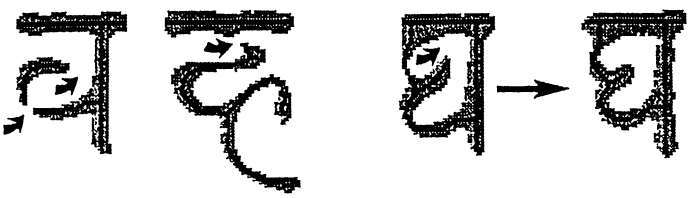
5.3 Error Analysis

In Table. 4, the errors have been divided into five categories. The final table gives the error percentage excluding the first two category errors. This is due to the fact that the errors from noise and breaks can be totally attributed to the printing defects in the text and are completely independent of the method used for recognition. Although the confusion between the characters 'pa' and 'ya' is also mainly due to the text, these two characters could have been distinguished through very stringent checks. This error is therefore included in the errors of the proposed method. The five categories and the problems caused by them are analysed below in detail with the help of examples.

1) Noise: The effect of noise in the recognition process has been of two types. The first case

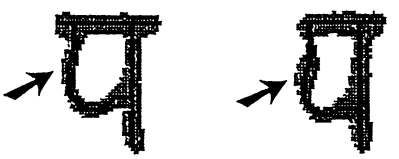
Correct 'a' Defective Correct 'śta' Defective

a. Noise in the character



'va' 'ha' 'tha' 'gha'

b. Break in the character



'pa' 'ya'

c. Similarity between 'pa' and 'ya'

'tta' Defective Correct

← 'nta' →

d. Printing defect leading to resemblance

Four categories of sources which causes erroneous results.

Fig. 11 Error analysis.

is where the noise is external to the character and the change is very basic, such as changing the size of the character and causing the basic checks to fail.

The second type is noise in the character. This has had the effect of changing the thinned result, so that the later checks, such as the checks of the number of free ends, have failed. This is illustrated in Fig. 11a.

2) Break: This again can be divided into two categories. When the break is in the main horizontal line or in any of the vertical lines, basic checks to detect these two fail and the character will be rejected without any detailed checks being performed. When the break occurs elsewhere in the character, the character is rejected at the lowest level; there is even a case where the break is such as to cause a resemblance to an otherwise different character; the examples are given in Fig. 11b.

3) Thinning defects: These have been examined in detail in section 3.

4) Confusion between 'pa' and 'ya': As can be seen from Fig. 11c, these two characters resemble each other to a great extent, though the examples given in the figure are extreme cases of similarity. The methodology adopted for identification was to scan the portion immediately below the MHL and at the left end, indicated by the arrows in the figure. The curvature of this portion was checked by finding the difference in pixels between the left character boundary and the most concave part of this portion. If this value was only 1 or 2, the character was supposed to be 'pa', else 'ya'. It can be seen from the figure that the original characters show more difference than the thinned characters.

5) Others: This category includes all the other errors, which again are due to two reasons. The first being a resemblance between two characters, especially after thinning, as the original character is blurred due to noise or other printing defects (Fig. 11d). The second category is due to insufficient decision making in the program.

6. Conclusion

Recognising Devanāgarī characters by analysing their features has had a promising beginning. The results obtained from the first phase of the research have also been encouraging.

Although the present system has been designed with the font of a particular text in mind, the same methodology can be extended to cover other fonts, especially the more recent ones. With the current state of printing technology, the results for these are expected to be much better. The next step would be to apply this method to hand-written characters, and finally, use linguistic knowledge to improve the results.

Acknowledgements

The authors wish to thank Asst. Professor M. Yamazaki of Sendai Technical College for help in reading the Buddhist text. They are also very glad to Mr. Paul Hoornat for his careful reading of the manuscript.

References

1) A. Suzuki, H. Kanai, S. Makino, Y. Kawazoe and K. Kido: Devanāgarī Character Recognition Method by Using Extraction and Recognition Procedures Simultaneously, *Journal of IEICE Japan*, Vol. J72-D-II, No. 10, pp. 1643-1649 (1989).

2) R.M.K Sinha and H.N. Mahabala: Machine Recognition of Devanāgarī Script, *IEEE Trans. on Systems Man and Cybernetics*, Vol. SMC-9, No. 8, pp. 435-441 (1979).

3) H.Kern and Bunyiu Nanjio (ed.): *Saddharmapuṇḍarīka*, St. Petersbourg, Imprimerie de L'Academie Imperiale des Sciences (1912).

4) C.M. Holt, A. Stewart, M. Clint and R.H. Perrott: An Improved Parallel Thinning Algorithm, *Commun. of the ACM*, Vol. 30, No. 2, pp. 156-160 (1987).

5) S. Mori, K. Yamamoto and M. Yasuda: Research on Machine Recognition of Handprinted Characters, *IEEE trans. on Pattern Analysis and Machine Intelligence*, Vol. PAMI-6, No. 4 (1984).

(Received January 6, 1990)