Sādhanā Vol. 27, Part 1, February 2002, pp. 23–34. © Printed in India

Automatic recognition of printed Oriya script

B B CHAUDHURI, U PAL and M MITRA

Computer Vision and Pattern Recognition Unit, Indian statistical Institute,
203, B T Road, Kolkata 700 108, India
e- mail: {bbc, umapada, mandar}@isical.ac.in

Abstract. This paper deals with an Optical Character Recognition (OCR) system for printed *Oriya* script. The development of OCR for this script is difficult because a large number of character shapes in the script have to be recognized. In the proposed system, the document image is first captured using a flat-bed scanner and then passed through different preprocessing modules like skew correction, line segmentation, zone detection, word and character segmentation etc. These modules have been developed by combining some conventional techniques with some newly proposed ones. Next, individual characters are recognized using a combination of stroke and run-number based features, along with features obtained from the concept of water overflow from a reservoir. The feature detection methods are simple and robust, and do not require preprocessing steps like thinning and pruning. A prototype of the system has been tested on a variety of printed Oriya material, and currently achieves 96.3% character level accuracy on average.

Keywords. Indian script; Oriya text; character segmentation; skew detection; optical character recognition (OCR).

1. Introduction

The subject of character recognition has been receiving considerable attention in recent years due to the advancement of the automation process. Automatic character recognition improves the interaction between man and machine in many applications like office automation, cheque verification, mail sorting, and a large variety of banking, business and data entry applications.

Several methods for recognizing Latin, Chinese, and Arabic script have been proposed in the recent past (Mantas 1986; Bozinovic & Srihari 1989; Wang 1991; Mori *et al* 1992) and commercial OCR systems for these scripts are available in the market. Some pioneering work has been done on Bengali (Bangla) (Dutta & Chaudhuri 1993; Chaudhuri & Pal 1997, 1998) and Devnagari (Sinha 1985; Pal & Chaudhuri 1997) and OCR systems for these scripts are ready for commercialization. Some studies have been reported on Tamil, Telugu and Gurmukhi scripts (Govindan & Shivaprasad 1990; Lehal & Singh 2000; Siromony *et al* 1978). However, to the best of our knowledge, no work has been done on the Oriya script. In this paper, we are concerned with the recognition of printed Oriya script. In Oriya, the number of characters is large, and two or more characters may combine to form complex shaped characters called *compound* or *clustered characters*. As a result, the total number of characters to be recognized is more than 200. Thus, OCR development for Oriya is more difficult than that for any European language script where a relatively small number of characters have to be recognized.

In the proposed recognition system, the document image is first captured using a flatbed scanner. The image is then passed through different preprocessing modules like skew correction, line segmentation, zone detection, word and character segmentation etc. These modules have been developed by combining conventional and newly proposed techniques. Next, individual characters are recognized using a combination of stroke and run-number based features, along with features obtained from the concept of *water overflow from a reservoir*.

The organization of this paper is as follows. In § 2, some properties of Oriya script are presented. Text digitization and skew correction techniques are described in § 3. Section 4 deals with line, word and character segmentation. Feature selection and detection are elaborated in § 5. Section 6 describes the character recognition procedure. Results are discussed in § 7, while § 8 concludes the paper.

2. Properties of Oriya script

We describe here some properties of the Oriya script that are useful for building the recognition system.

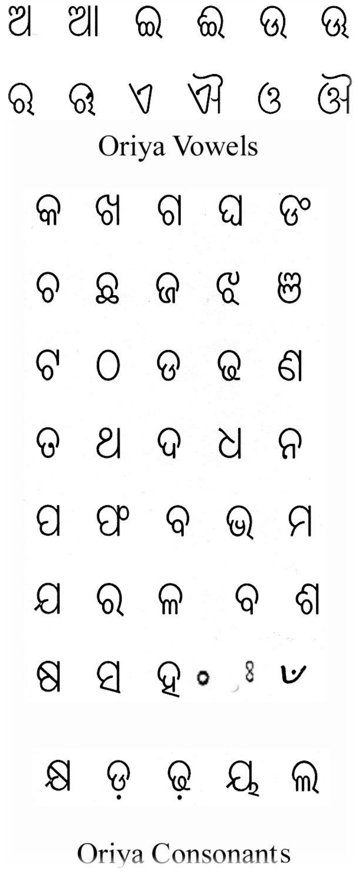

Figure 1. Basic characters of the Oriya alphabet.

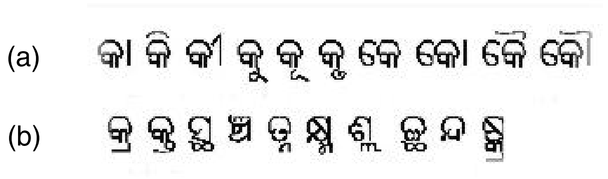

Figure 2. **(a)** Modified vowels attached to the first consonant of figure 1. **(b)** Some commonly occurring compound characters.

- The Oriya script is derived from the ancient Brahmi script through various transformations. Other Indian scripts also have the same origin, making some of them similar in appearance to Oriya.
- There are 12 vowels and 39 consonants in the modern Oriya alphabet. These are called *basic* characters. The basic characters of Oriya Script are shown in figure 1. Their names are identical to the names for corresponding characters in other scripts like Bengali and Devnagari. As in other Indian scripts, the concept of upper/lower case is absent here.
- Some characters have a signature extending above the mean line, which is also useful for character classification.
- The first vowel following a consonant character is not printed in a word. Thus, this vowel can occur only at the beginning of a word.
- Vowels (other than the first one) following a consonant take some modified shapes. Depending on the vowel, these modified shapes are placed to the left, right (or on both sides), top or bottom of the consonant. These are called modified characters or *allographs*. See figure 2a, where vowel modifiers are shown. From the figure it can be seen that some modified shapes of vowel may have two parts – one appears to the left and the other to the right of a consonant (or compound character). More precisely, modifiers are those symbols that do not disturb the shape of the basic characters (in the middle zone) to which they are attached. If the shape in the middle zone is altered, we term the resultant shape as a compound character. Some of the compound character shapes are shown in figure 2b.
- In some cases, a consonant preceding or following another consonant is represented by a modifier called consonant modifier.
- A word in Oriya script may be partitioned into three zones: upper zone, middle zone and lower zone. The upper zone denotes the portion above the mean line, the middle zone covers the region below the mean line, and the lower zone is the portion where some of the modifiers can reside. The imaginary line separating the middle and lower zone is called the base line. A typical example of zoning is shown in figure 3.

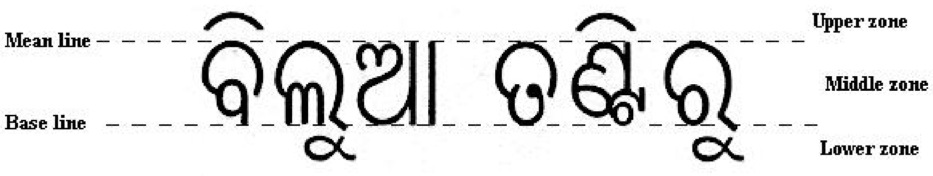

Figure 3. Zones in an Oriya text line.

3. Text digitization and skew correction

3.1 *Text digitization and noise cleaning*

Text digitization is done using a flatbed scanner (Model: HP Scanjet 660 C) at a resolution varying from 200 to 300 dots per inch (dpi). The digitized images are in gray tone and we have used a histogram-based thresholding approach to convert them into two-tone images. For a clear document, the histogram shows two reasonably prominent peaks corresponding to white and black regions. The threshold value is chosen as the midpoint between the two histogram peaks. The two-tone image is converted into 0–1 labels where 1 and 0 represent object and background respectively. The digitized image shows protrusions and dents in the characters, as well as isolated black pixels over the background, which are cleaned by a logical smoothing approach (Chaudhuri & Pal (1998)).

3.2 *Skew detection and correction*

When a document is fed to the scanner either mechanically or by a human operator, a few degrees of skew (tilt) is unavoidable. The *skew angle* is the angle that the text lines in the digital image make with the horizontal direction. Skew detection and correction are important preprocessing steps of document layout analysis and OCR approaches. Skew correction can be achieved in two steps, namely (i) estimation of skew angle, and (ii) rotation of the image by the skew angle in the opposite direction.

There exist many techniques for skew estimation. One skew estimation technique is based on the projection profile (Akiyama & Hagita 1990; Pavlidis & Zhou 1992) of the document. The horizontal/vertical projection profile is a histogram of the number of black pixels along horizontal/vertical scan-lines. For a script with horizontal text lines, the horizontal projection profile will have peaks at text line positions and troughs at positions in between successive text lines. To determine the skew angle of a document, the projection profile is computed at a number of angles, and for each angle, the difference between peak and trough heights is measured. The maximum difference corresponds to the best alignment with the text line direction. This in turn determines the skew angle.

Another class of approaches is based on nearest neighbour clustering of connected components. O'Gorman (1993) proposed one nearest neighbour clustering method, called 'docstrum', for skew detection. He detected all the connected components in the document and for each component computed the direction of its nearest neighbour. A histogram of the direction angle is computed, the peak of which indicates the document skew angle. Chaudhuri & Pal (1997) proposed a generalized clustering approach for skew detection of Indian script documents. They used the mean line of the script for skew estimation.

Techniques based on the Hough transform and Fourier transform are also employed for skew estimation (Hinds *et al* 1990; Le *et al* 1994). In our work, we used a Hough transform based technique for estimating the skew angle of Oriya documents. We note that the uppermost and lowermost points of most of the characters in an Oriya text line lie on the mean line and base line respectively. The lowermost and uppermost points of characters in a skewed Oriya text are shown in figure 4. To reduce the amount of data to be processed by the Hough transform, we consider only the uppermost and lowermost pixels of each component. First, the connected components in a given image are identified. For each component, its bounding box (minimum upright rectangle containing the component) is defined. The mean width of the bounding boxes b_m is also computed. Next, components having bounding box width greater than or equal to b_m are retained. By thresholding at b_m, small components like dots,

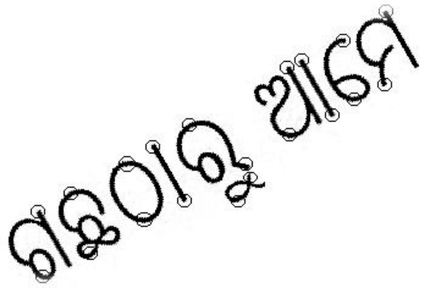

Figure 4. Uppermost and lowermost points of components in a skewed text line.

punctuation marks, small modified characters etc. are mostly filtered out. Because of this filtering process, the irrelevant components cannot create errors in skew estimation. Now, the usual Hough transform technique is used on these points to get the skew angle of the document. The image is then rotated according to the detected skew angle. Font style and size variation do not affect the proposed skew estimation method. Also, the approach is not limited to any range of skew angles.

4. Line, word and character segmentation

For convenience of recognition, the OCR system should automatically detect individual text lines, segment the words from the line, and then segment the characters in each word accurately. Since Oriya text lines can be partitioned into three zones (see figure 3), it is convenient to distinguish these zones. Character recognition becomes easier if the zones are distinguished because the lower zone contains only modifiers and the *halant* marker, while the upper zone contains modifiers and portions of some basic characters.

4.1 *Text line detection and zone separation*

The lines of a text block are segmented by finding the valleys of the projection profile computed by counting the number of black pixels in each row. The trough between two consecutive peaks in this profile denotes the boundary between two text lines. A text line can be found between two consecutive boundary lines, for example, see figure 5. We have assumed that the text block contains only a single column of text.

After line segmentation, the zones in each line are detected. From figure 3 it can be seen that the upper zone is separated from the middle zone of a text line by the mean line, and

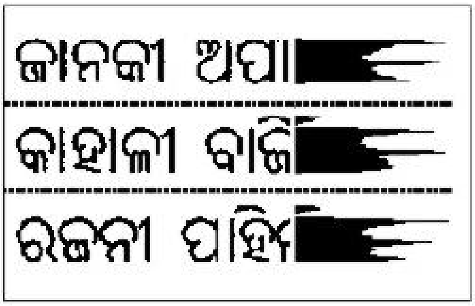

Figure 5. Projection profile of rows in Oriya text line (dotted lines show line boundaries).

that the middle zone is separated from the lower zone by the base line. We use the uppermost and lowermost points of the connected components in a text line to detect the mean line and base line respectively. We consider a set of horizontal lines passing through the uppermost and lowermost points of the components. The horizontal line that passes through the maximum number of uppermost points (lower most points) is the mean line (base line). Note that the uppermost and lowermost points of the components are previously detected during skew detection (see § 3), so these points do not have to be recalculated during zone detection.

4.2 *Word and character segmentation*

After a text line is segmented, it is scanned vertically, column by column. If a column contains two or fewer black pixels, the scan is denoted by 0, else it is denoted by the number of black pixels in that column. In this way, a vertical projection profile is constructed. Now, if in the profile there exists a run of at least k_1 consecutive 0s, the midpoint of that run is considered the boundary between two words. The value of k_1 is taken as two-thirds of the text line height (text line height is the normal distance between the mean line and the base line).

To segment each word into individual characters, we consider only the middle zone of the word. To find the boundary between characters, we scan the image of the word in the vertical direction starting from the mean line. If during a scan, we reach the base line without encountering any black pixel, this scan marks the boundary between two characters. However, the gray-tone to two-tone conversion of the image gives rise to some characters that touch one another, and which cannot be segmented using this method. To segment these touching characters, we have used the principle of water overflow from a reservoir, which is as follows. If we pour water on top of the character, the positions where water will accumulate are considered the reservoirs. Figure 6 shows the location of reservoirs in a single character as well as in a pair of touching characters. We note the height of the water level in the reservoir, the direction of water overflow from the reservoir, position of the reservoir with respect to the character bounding box etc. A reservoir whose height is small and which lies in the upper part of the middle zone of a line is considered a candidate reservoir for touching character segmentation. The cusp (lowermost point) of the candidate reservoir is considered the separation point of the touching characters. In figure 6, this position is marked by a vertical line. Because of the round shape of most of the Oriya characters, we observe that such a reservoir is formed in most of the cases when two characters touch each other. Sometimes, two or more reservoirs may be formed. In such cases, we select the reservoir close to the middle of the bounding box for segmentation.

5. Feature selection and detection

We consider topological features, stroke-based features as well as features obtained from the concept of water overflow for character recognition. We term these the principal features. The features are chosen with the following considerations: (a) Robustness, accuracy and simplicity of detection, (b) speed of computation, (c) independence of size and fonts, and (d) tree classifier design need.

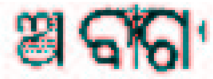

Figure 6. Water reservoirs in a single and touching Oriya characters.

We considered a few stroke-based and topological features for the initial classification of characters. These features are used to design a tree classifier where the decision at each node of the tree is taken on the basis of the presence/absence of a particular feature. Stroke-based features include the number and position of vertical lines. The topological features used include existence of holes and their number, position of holes with respect to the character bounding box, ratio of hole height to character height etc. In addition, the concept of water overflow from a reservoir is also used. The reservoirs in a character are identified (see § 4), and the position of the reservoirs with respect to the character bounding box, the height of each reservoir, the direction of water overflow etc. are used as features in the recognition scheme.

The stroke features considered here are simple, linear in structure, and hence quick and easy to detect. They are fairly robust to noise and quite stable with respect to font variation. The methods for detecting the stroke features are described below. To handle characters of different fonts and sizes, the stroke lengths are normalized with respect to the character middle-zone height.

To detect vertical strokes, we assume that the length of the stroke must be at least $l_1 (l_1 =$ 75% of the middle-zone height). Now, the character middle zone is scanned vertically. If a scan contains a continuous sequence of at least l_1 black pixels, a vertical stroke is assumed to be present in that character. For the detection of holes, a component labelling technique is used. The background and foreground values in the character image are interchanged, and connected components are identified in this modified image. Each hole then constitutes a separate connected component. The width and height of such topological features are also measured with respect to the height of a text line.

6. Character recognition

The recognition stage has two parts. In the first part, modified characters are recognized and, in the second part, the remaining characters are recognized. Modified characters are distinguished from the other characters by making use of the width and the position of the characters. If the width of a character is very small, or if the position of the character is only in the upper zone or only in the lower zone, the character is considered a modified character.

Our recognition scheme for modifiers is based on stroke-based features and a run-number based feature. Any isolated signature in the upper zone that does not touch a vertical line in the middle zone denotes the modifier for the upper zone. To detect lower zone modifiers, the lower zone is inspected. If there is any signature in the lower zone that is not a part of a basic or compound character in the middle zone, it is assumed to be a lower zone modifier. Three such modifiers and the halant sign may be encountered in this region. To check if a signature in the lower zone is actually the tail of a basic character, a curvature-smoothing test is used (Chaudhuri & Pal 1998). If the signature curvature is continuous and smooth, it is accepted as the tail of a basic character.

Recognition of the basic and compound characters is done in two stages. In the first stage, the characters are grouped into small subsets by a feature based tree classifier. In the second stage, characters in each group are recognized using a sophisticated run-number based matching approach. We adopted this hybrid approach instead of using only a tree classifier because it is nearly impossible to find a set of stroke features that are simple to compute, robust and reliable to detect, and are sufficient to classify a large number of basic and complex shaped compound characters.

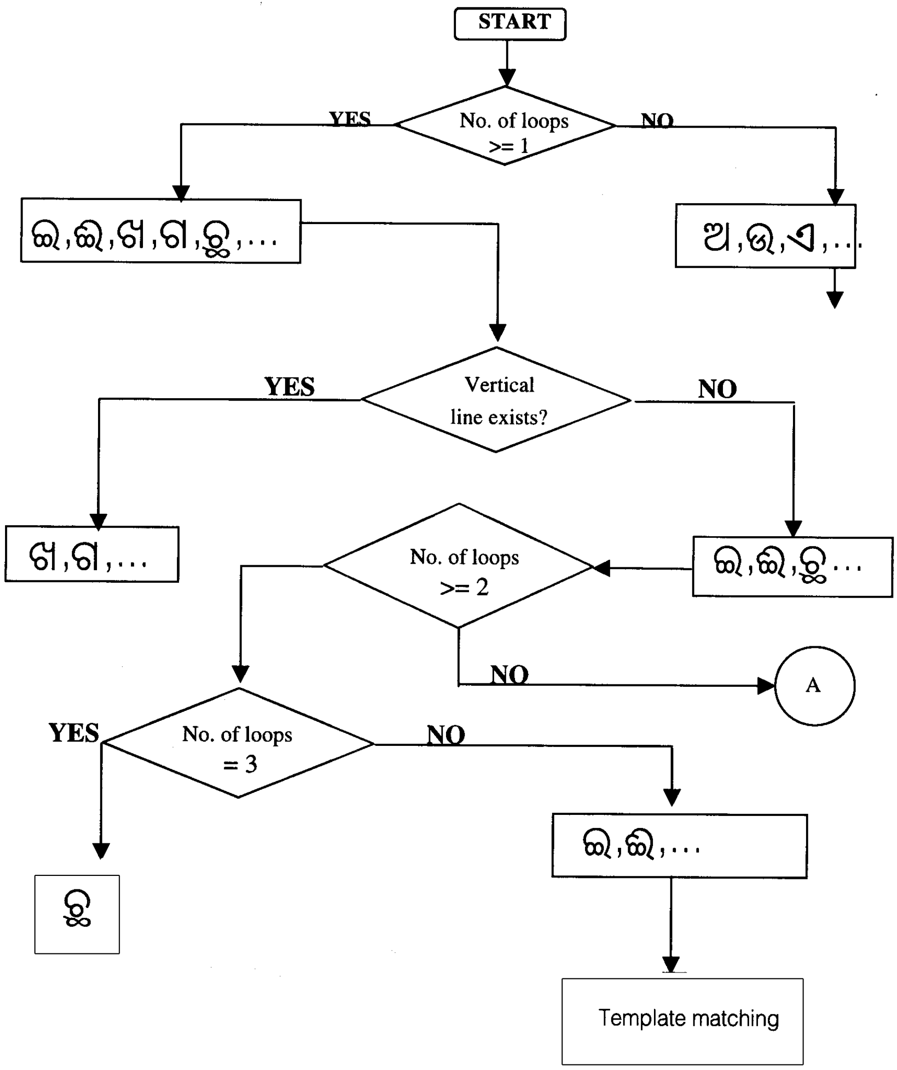

Figure 7. Portion of the tree classifier for Oriya characters.

The design of a tree classifier has three components: (1) a tree skeleton or hierarchical ordering of the class labels, (2) choice of features at each non- terminal node, and (3) the decision rule at each non-terminal node. Our tree is a binary tree where the number of descendants from a non-terminal node is two. While traversing the tree, only one feature is tested at each non-terminal node. To choose the feature at a particular non-terminal node, we have considered the occurrence statistics of the characters. If the set of patterns at a non-terminal node can be sub-divided into two sub-groups by examining a feature so that the sum of occurrence probabilities of one group is roughly equal to that of the other group, the resulting binary tree is optimum in time complexity, assuming that the time required to test a feature is constant. However, we may not always get a set of features to design such an optimal tree. A semi-optimal tree is generated out of the available features. For a given non-terminal node, we select a feature that best separates the group of patterns in the above sense.

A part of the tree classifier used for the recognition of basic and compound characters is shown in figure 7. The features used in the tree classifier are vertical strokes, holes, features based on the concept of water overflow from reservoirs etc. The use of some of these features can be noted from the classification tree. Figure 7 does not contain the entire tree, however. For example, in a portion of the tree not shown in the figure, we check whether the height of a hole is equal to the line height, and separate nine characters of a node into two sub-nodes, one of which contains four characters and the other contains five. Similarly, by checking whether a hole (reservoir) touches the mean line of the character or not, we can separate five characters of a node into two sub-nodes containing three and two characters.

Most leaf nodes of the classification tree contain two or more characters. At the leaf nodes, we use a run-number based feature matching technique (Garain & Chaudhuri 1998) in order to distinguish between characters. The run-number based feature is obtained as follows. We compute the number of black runs (a run is a contiguous sequence of black pixels) in each row of the character. If there are n rows, this yields a sequence of n numbers, $c_1, \cdots, c_n$. This sequence is then run-length encoded to obtain the sequence $\langle c_1, n_1\rangle, \cdots \langle c_k, n_k\rangle$, where the first n_1 rows contain c_1 runs, the next n_2 rows contain c_2 runs, and so on. Note that $n_1+n_2+\cdots+n_k = n$. The number of rows corresponding to each run count is then normalized using the total number of rows to obtain the final row-feature vector for the character:

$$\langle c_1, r_1\rangle, \cdots \langle c_k, r_k\rangle \text{ where } r_i = n_i/n. \tag{1}$$

A column-feature vector is similarly computed using the number of black runs in each column of the character image.

The distance between two feature vectors is calculated using the following method. Given two vectors $A = \langle a_1, u_1\rangle, \cdots \langle a_k, u_k\rangle$ and $B = \langle b_1, v_1\rangle, \cdots \langle b_l, v_l\rangle$, we define

$$\begin{aligned} A' &= \langle a_1, u'_1\rangle, \cdots, \langle a_k, u'_k\rangle \text{ where } u'_i = \sum u_j (j = 1 \text{ to } i), \\ B' &= \langle b_1, v'_1\rangle, \cdots, \langle b_l, v'_l\rangle \text{ where } v'_i = \sum v_j (j = 1 \text{ to } i), \end{aligned} \tag{2}$$

We then sort the union of $\{u'_1, \cdots, u'_k\}$ and $\{v'_1, \cdots, v'_l\}$ together in increasing order. Let the sorted sequence be $\{w_1, \cdots, w_m\}$ (where $w_m = 1$). Then the interval w_{i-1} to w_i is a (possibly proper) sub-interval of some interval u'_{j-1} to u'_j. Note that this interval corresponds to a sequence of n_j rows (where $u_j = n_j/n$) for which the run-count is a_j. Thus, the interval w_i to w_{i-1} can be associated with this run count a_j. A distance measure between the vectors can now be defined as

$$\text{Dist } (A, B) = \sum (w_i - w_{i-1})|a_i - b_i|, \quad i = 1 \text{ to } m, (\text{take } w_0 = 0), \tag{3}$$

(see Garain & Chaudhuri 1998) for a proof that this distance measure satisfies the properties of a metric distance). Finally, the dissimilarity between two characters is given by the sum of the distances between the row and column feature vectors for these characters.

7. Results and discussion

The proposed OCR system was tested on a variety of printed Oriya documents. Some of the documents contain good-quality printing on clean paper (e.g. pages from a novel); some others are of inferior printing and paper quality (e.g. a cheap alphabet book for children) etc. In this section, we summarize the results of our experiments.

Line segmentation: Our system identifies individual text lines with an accuracy of 97.5%. Occasionally, when two adjacent text lines are close to each other, the lower zone of the upper line has some overlap with the upper zone of the lower line. In such situations, there is no clear valley between the two lines in the projection profile, and our system fails to detect the boundary between the two lines. All such errors were confined to inferior-quality documents however; on good-quality documents, the system correctly identifies all text lines. We intend to try a connected component based approach for line segmentation in future.

Word segmentation: The overall word segmentation accuracy of the system is 97.7%. The error rate for the inferior documents is 4.2%, whereas for good-quality documents, it is only 1.2% (these figures were calculated based on correctly segmented text lines only). The threshold value chosen for the inter-word gap ($k_1 = 2/3$ of the text line height; see § 4) works well in most cases. However, because of non-uniform printing, some words are printed closer together and are not correctly separated by our system.

Character segmentation: The character segmentation accuracy of the system is 97.2%. The proposed method for separating touching characters based on the water reservoir concept is generally successful. Most of the segmentation errors were caused by the modified form of the fourth vowel (long I). As seen in figure 2a, this allograph has a small slanting projection at the top that often touches the adjacent character. Since no proper reservoir is obtained in this case, our method fails to correctly separate such touching characters.

Character recognition: On average, the system recognizes characters with an accuracy of about 96.3%, i.e. the overall error rate is 3.7%. The recognition errors can be grouped into three major classes.

(1) Errors due to segmentation. Incorrectly segmented characters can obviously not be recognized correctly. These errors contribute 0.4% to the total error rate. (The set of images for which recognition accuracy was computed is a subset of the images used to calculate the accuracy of character segmentation. There is thus a small difference between the figures in this and the preceding paragraph.)

(2) Placing a character in the wrong leaf of the classification tree. Though the features used in the classification tree are robust, a character is occasionally classified into the wrong leaf node, giving rise to recognition errors. The contribution of these errors to the overall error rate is only 0.4% however.

(3) Errors during feature matching. Though the run-number based feature is powerful and insensitive to font size and style variations, it may not always correctly distinguish two similar shaped characters. Thus, a character may sometimes be mis-recognized as another character contained in the same leaf node of the classification tree. Some examples of characters that have very similar appearances are shown in figure 8. This class of errors contributes 2.7% to the overall error rate. (Other errors contribute the remaining 0.2% to the total error rate.)

To recognize such characters properly, we propose to use border-pixel tracing. For example, for the top left pair shown in figure 8, we can start from the topmost pixel of the character and trace the border pixels in the clockwise direction until we reach the lowermost point of the character. During border tracing, we calculate the distance of each traced pixel from the right edge of the character's bounding box. The sequence of these distances can be used to distinguish the characters. For example, for the first character, this distance sequence has 4 transition points, while for the second character, we get 2 transition points. By a transition point, we mean a point where the sequence of distances changes from increasing order to

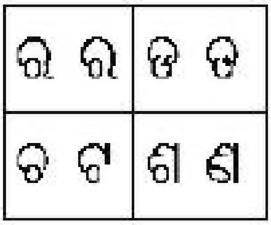

Figure 8. Some similar shaped Oriya characters.

decreasing order or vice-versa. This technique can be used for the recognition of some of the other similar characters also.

8. Conclusion

In this paper, we have described a system for OCR of printed Oriya script material. The recognition accuracy of the prototype implementation is promising, but more work needs to be done. In particular, no fine-tuning of the system has been done so far. Our character segmentation method also needs to be improved so that it can handle a larger variety of touching characters, which occur fairly often in images obtained from inferior-quality printed material. We also need to test the proposed boundary-tracing method for distinguishing between characters that have very similar shapes. Finally, we plan to implement an OCR error detector and corrector module for Oriya. In general, the system needs to be tested on a wider variety of images containing characters in diverse fonts and sizes. This will enable us to identify the major weaknesses in the system and implement remedies for them.

As a native speaker of Oriya, Anil Chand gave us useful advice about the script. P. Sashank helped with the implementation and testing of some of the algorithms proposed in this paper. The authors would also like to thank Vidya Dutt, Shamita Ghosh, and Prasenjit Majumdar for their help in preparing this manuscript.

References

Akiyama T, Hagita N 1990 Automatic entry system for printed documents. *Pattern Recogn.* 23: 1141–1154

Bozinovic R M, Srihari S N 1989 Off line cursive script word recognition. *IEEE Trans. Pattern Anal. Machine Intell.* PAMI-11: 68–83.

Chaudhuri B B, Pal U 1997 Skew angle detection of digitized Indian script documents, *IEEE Trans. Pattern Anal. Machine Intell.* PAMI 19: 182–186

Chaudhuri B B, Pal U 1997 An OCR system to read two Indian language scripts: Bangla and Devnagari (Hindi). *Proc. Fourth Int. Conf. on Document Analysis and Recognition* (Los Alamitos, CA: IEEE Comput. Soc.) pp 1011–1016

Chaudhuri B B, Pal U 1998 A complete printed Bangla OCR system. *Pattern Recogn.* 31: 531–549

Dutta A K, Chaudhuri S 1993 Bengali alpha-numeric character recognition using curvature features. *Pattern Recogn.* 26: 1757–1770

Garain U, Chaudhuri B B 1998 Compound character recognition by run number based metric distance. *Proc. SPIE Annual Symposium on Electronic Imaging*, San Jose, USA, pp 90–97

Govindan V K, Shivaprasad A P 1990 Character recognition - a survey. *Pattern Recogn.* 23: 671–683

Hinds S C, Fisher J L, D'Amato D P 1990 A document skew detection method using run-length encoding and the Hough transform. *Proc. 10th Int. Conf. on Pattern Recognition* (Los Alamitos, CA: IEEE Comput. Soc.) vol. 1, pp 464–468

Le D S, Thoma G R, Wechsler H 1994 Automatic page orientation and skew angle detection for binary document images. *Pattern Recogn.* 27: 1325–1344

Lehal G S, Singh C 2000 A Gurmukhi script recognition system. *Proc. 15th Int. Conf. on Pattern Recognition* (Los Alamitos, CA: IEEE Comput. Soc.) vol. 2, pp 557–560

Mantas J 1986 An overview of character recognition methodologies. *Pattern Recogn.* 19: 425–430

Mori S, Suen C Y, Yamamoto K 1992 Historical review of OCR research and development. *Proc. IEEE* 80: 1029–1058

O'Gorman L 1993 The document spectrum for page layout analysis. *IEEE Trans. Pattern Anal. Machine Intell.* PAMI-15: 1162–1173

Pal U, Chaudhuri B B 1997 Printed Devnagari script OCR system. *Vivek* 10: 12–24

Pavlidis T, Zhou J 1992 Page segmentation and classification. *Comput. Vision Graphics Image Process.* 54: 484–496

Sinha R M K 1987 Rule based contextual post processing for Devnagari text recognition. *Pattern Recogn.* 20: 475–485

Siromony G, Chandrasekaran R, Chandrasekaran M 1978 Computer recognition of printed Tamil characters. *Pattern Recogn.* 10: 243–247

Wang P S P 1991 *Character and handwritten recognition* (Singapore: World Scientific)