Segmentation of Touching Characters in Printed Devnagari and Bangla Scripts Using Fuzzy Multifactorial Analysis

Utpal Garain and Bidyut B. Chaudhuri, *Fellow, IEEE*

***Abstract*—One of the important reasons for poor recognition rate in optical character recognition (OCR) system is the error in character segmentation. Existence of touching characters in the scanned documents is a major problem to design an effective character segmentation procedure. In this paper, a new technique is presented for identification and segmentation of touching characters. The technique is based on fuzzy multifactorial analysis. A predictive algorithm is developed for effectively selecting possible cut columns for segmenting the touching characters. The proposed method has been applied to printed documents in Devnagari and Bangla: the two most popular scripts of the Indian sub-continent. The results obtained from a test-set of considerable size show that a reasonable improvement in recognition rate can be achieved with a modest increase in computations.**

***Index Terms*—Fuzzy decision making, Indian script optical character recognition (OCR), multifactorial analysis, touching characters.**

I. Introduction

DOCUMENT analysis systems facilitate transfer of information on a paper document to the computer systems without intensive manual keying. For certain language scripts (e.g., Roman script) today, it is not difficult to develop an optical character recognition (OCR) system that recognizes well- shaped and well-spaced characters with accuracy of 99% and above. However, it is still challenging to design a system that can maintain such high recognition accuracy, regardless of the quality of the input document and character font style variation.

In order to recognize the text contained in a document, it is usually segmented into lines, words, and characters. Then, the characters are recognized using a two-step process: i) extracting the distinguished features from the character image and ii) finding the member of a predefined symbol set whose features best match those of the input. In this procedure, researchers have noted that character segmentation errors, as discussed in [1]–[4] contribute heavily to the OCR error rate. The well-known tests of commercial printed text OCR systems conducted by University of Nevada, Las Vegas [5], also support this observation. In many occasions, the adjacent characters touch each other in the scanned image, and separation of such touching characters is a major problem. This is because the text is typically segmented by connectivity analysis, which considers touching characters as a single unit.

This paper is concerned with the problems related to touching characters. Fujisawa *et al.* [3], Elliman and Lancaster [6], Casey and Lecolinet [7], have presented elaborate surveys on character segmentation. These studies are mainly aimed at character separation for OCR, but some references contain discussion about segmentation of touching characters as well. From these references, we find two categories of approach, where i) the touching character segmentation and recognition go hand-in-hand, or ii) the recognition is attempted without segmentation (or segmentation is implicit in nature).

Some detailed concept of the first category may be found in [1] and [8]. Here, the basic principle is to generate the segmentation hypothesis based on some objective (or discrimination) function and then to choose the best hypothesis. In this process, multiple sequences (or cut columns) are obtained from the touching character patterns, and each sequence is assessed based on the recognition result. The method reported by Liang *et al.* [9] was also based on the strategy proposed in [1]. However, they used a discriminating function that was different from the one used in [1]. The approach proposed by Bayer and Kressel [10] was also based on the technique of generating and testing the cut hypothesis for segmentation of touching characters with the difference that the decision rules were created automatically rather than by man-made heuristics. Kahan *et al.* [11] used projection-profile analysis where the curve for the vertical projection was first obtained, and then, the ratio of the second derivative of the curve to its height was used as an objective function. Later on, their method was improved by introducing a peak-to-valley function [12]. A prefiltering was implemented by Tsujimoto and Asada [2] in order to intensify the function.

Among others, Rocha and Pavlidis [13] presented a different approach where character segmentation is implicit in nature. The method attempts to match sub-graphs of features with predefined character prototypes. Different alternatives are represented by a network whose nodes correspond to the matched sub-graphs.

Most of the techniques discussed above deal with Roman scripts and, to the best of our knowledge, no similar study is available for Indian language documents. This paper is an attempt to fill that gap. The development of OCR systems for the Indian language scripts has recently been a serious area of research because of its large market potential. As a result, a

Manuscript received August 28, 2001; revised April 21, 2002. A shorter and preliminary version of this paper was presented in *International Conference on Document Analysis and Recognition* (ICDAR), Seattle, WA, USA, 2001.

The authors are with the Computer Vision and Pattern Recognition Unit, Indian Statistical Institute, Kolkata 700 108, India (e-mail: utpal@isical.ac.in; bbc@isical.ac.in).

Digital Object Identifier 10.1109/TSMCC.2002.807272

Bangla OCR system [14] and, later on, a bilingual (Devnagari and Bangla) OCR system [15], [16] has been developed. The motivation for developing this bilingual OCR was that since both Bangla and Devnagari are derived from the same ancient script Brahmi, they have many features in common. The character names in the alphabet are the same, only their graphemic shapes are different. Devnagari is the most popular script on the Indian sub-continent and is used to write Hindi (the most popular Indian language) as well as Sanskrit, Marathi, Sindhi, and Nepali language with minor modifications. On the other hand, Bangla, Assamese, and Manipuri languages are written in Bangla script. In total, more than 700 million people use these languages in India, Bangladesh, and Nepal.

The work described in this paper is a part of our OCR project on Devnagari and Bangla. Our original system yields recognition accuracy of about 98% for documents of good print and paper quality. However, it shows poor performance for documents like

1) old books: print and paper quality inferior due to aging;
2) copied materials: documents like photocopies or faxed documents, where print quality is inferior to the original;
3) newspapers: generally printed on low-quality paper, etc.

For such degraded documents, the system recognition accuracy comes down to 85–90%. We observed that the character segmentation phase is crucial in contributing to this error. It is statistically surveyed [17] that Hindi and Bangla documents from old books, photocopies, or faxed documents, newspapers, etc. contain 5 to 13% touching characters, which the classifier cannot properly tackle. Even in good-quality documents, some adjacent characters touch each other due to inappropriate scanning resolution.

To tackle the touching characters in Devnagari and Bangla documents, at first, we attempt to identify the touching characters. Next, they are segmented into constituent ones using a fuzzy decision-making approach. A typical set of touching characters in the Devnagari and Bangla scripts is shown in Fig. 4. Sometimes, it is difficult even for humans to isolate the characters from certain touching patterns, and the touching process may be considered as a fuzzy phenomenon. Moreover, the selection of appropriate cut columns (i.e., segmentation points) depends on many factors, making it a multifactorial decision-making problem. So, the framework for fuzzy multifactorial analysis is used to segment the touching characters in Devnagari and Bangla documents.

In our approach, we do not crisply select a pixel column for segmenting touching characters. Rather, we assign fuzzy membership to all the columns. To assign the membership, we evaluate each column against a number of factors on which the selection of cut column depends. For this multifactorial evaluation, we use additive standard multifactorial (ASM) functions [18]. Once this evaluation is done, the pixel columns in the image of the touching characters are then sorted in descending order according to their degree of membership.

These membership values are used to predict the right cut column(s) to get the correct component character sequences from the touching characters. For this purpose, we use a concept similar to the concepts of predictive parser [19] used in compiler design for programming languages. Because of this technique, unlike the methods described in [1], [2], and [9], our approach substantially reduces the number of possible cut columns. Extensive studies with different text documents have attested to the feasibility of the proposed algorithm.

The rest of the paper is organized as follows. Section II discusses some of the properties of the Devnagari and Bangla script and the techniques for line, word, and character segmentation that are employed in our system [16]. In this section, we also look at the problem related to the origin and properties of touching characters. A general discussion on the fuzzy multifactorial analysis technique is presented in Section III. Section IV outlines identification of touching characters and their segmentation by the fuzzy multifactorial evaluation technique. The proposed approach is tested on a considerable number of different documents, and the results are presented in Section V. Section VI contains concluding remarks.

II. Devnagari and Bangla OCR Preprocessing and Touching Characters

Since Devnagari and Bangla are Asian scripts, we describe some of their properties for the benefit of wider readership. In both scripts, the writing style is horizontal, left to right, and the characters do not have any uppercase/lowercase distinction. There are 50 basic characters in both scripts having nearly one-to-one correspondence. Moreover, the corresponding characters are called by the same name. Among the characters, the vowels often take modified shapes in a word. Such characters are called modifiers or allographs. Consonant modifiers are also possible. Moreover, several consonant characters may combine to form compound characters that partly retain the shape of the constituent characters. Most compound characters are formed by joining two consonants, but consonant compounding of up to four is possible. Because of them, the number of characters to be recognized in each script is about 300.

The set of basic characters (with a small subset of compound characters) for both scripts are shown in Fig. 1. The number of compound characters in Bangla and Devnagari is more than 250. However, in both scripts, the compound characters occur less frequently (only 5 to 7%). For a large number of characters (basic as well as compound), there exists a horizontal line at the upper part called *shirorekha* in Hindi and *matra* in Bangla language. For our convenience, we call it *headline*. The characters of a word are actually connected through the headline (see Fig. 2). Many characters, including vowel modifiers, in both scripts have vertical strokes. The punctuation mark for period or full stop is also a vertical line.

In both scripts, a text word may be partitioned into three zones. The upper zone denotes the portion above the headline, the middle zone covers the portion of basic and compound characters below the headline, and the lower zone may contain where some vowel and consonant modifiers can reside. The imaginary line separating the middle and lower zone may be called the base line. A typical zoning is shown in Fig. 2.

Before we discuss the problem related to the touching characters, we briefly describe the preprocessing steps followed in the OCR system [16] developed by our group.

Fig. 1. Basic characters in (a) Devnagari and (b) Bangla scripts. First row contains vowels and the second and third rows contain consonants. A small subset of compound characters is shown in last two rows.

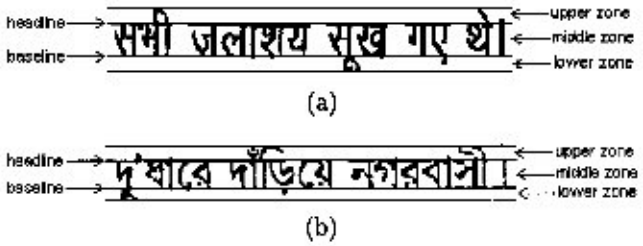


Fig. 2. Headline, baseline, and three zones in (a) Devnagari and (b) Bangla.

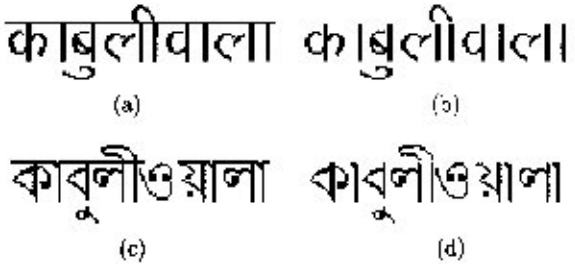


Fig. 3. Character segmentation through headline deletion. (a) Devnagari word. (b) Devnagari word after headline deletion. (c) Bangla word. (d) Bangla word after headline deletion.

A. Text Digitization and Preprocessing Techniques

In our system, text digitization is done by a flatbed scanner having resolution between 100 and 600 dpi. The digitized images are usually in gray tone, and for a clear document, a simple histogram-based thresholding approach is sufficient for converting them to two-tone images. The histogram of gray values of the pixels shows two prominent peaks, and a middle gray value located between the peaks is a good choice for the threshold.

When a document page is fed to the scanner, it may be skewed by a few degrees. The pages of a thick book create more problems since the scanned images may both be distorted and skewed. There exists a wide variety of skew detection algorithms based on projection profile, Hough transform, nearest neighbor clustering [20], docstrum analysis [21], line correlation [22], etc. Chaudhuri and Pal [23] proposed a new approach that employs the headline of Devnagari and Bangla script. In our system, skew detection and correction is done by the method described in [23].

B. Line, Word, and Character Segmentation

Once the text blocks are detected, the system automatically finds individual text lines, segments the words, and then separates the characters accurately. The text lines are detected by finding the valleys of the projection profile computed by a row-wise sum of black pixels. This profile shows large values for the headline of an individual text line. The position between two consecutive headlines, where the projection profile height is minimum, denotes the boundary between two text lines.

The words of a text line are distinguished by looking at the vertical pixel projection profile, where gaps between the words show up as reasonably long minima. Character segmentation is done after the individual words are identified. Since characters in a word are usually connected to each other through the headline, they get disconnected once the headline is erased. However, in our system, for segmenting the characters, a connected component analysis is implemented for the region just below the headline. Fig. 3(a) and (b) shows segmentation of a Devnagari word into its constituent characters. Fig. 3(c) and (d) shows the same for a Bangla word.

C. Problem of Touching Characters

The technique for character segmentation, described previously, treats characters touching each other as a single character and leads to the failure in the character recognition phase. Faxed documents, photocopies, old books, newspapers, etc., contain a considerable number of touching characters. The module for automatic separation of touching characters is essential for successful OCR of such documents.

For designing such a module, we carried out a study on relative abundance of touching characters in various printed Devnagari and Bangla documents. Sample documents were collected from leading newspapers, books published more than 20 years ago, faxed documents, photocopies, etc. The number of document pages taken from these sources and the frequency of touching characters found there are summarized in Table I. The table shows that the old books contain the highest percentage of touching characters. This is due to the font style, inferior printing, and paper quality as well as aging of documents. Some typical touching characters are shown in Fig. 4.

Next, we introduce some observations based on the statistical analysis of touching characters.

- *Observation 1:* In most cases, an image of touching characters contains two characters. Three or more characters touching each other are rare. More details are given in Table II.
- *Observation 2:* The combined shapes of touching characters are markedly different from any valid character (on the other hand, in Roman script, characters like "r" and "n," when they touch each other, look like a valid character "m").

TABLE I
RELATIVE ABUNDANCE OF TOUCHING CHARACTERS IN DEVNAGARI AND BANGLA DOCUMENTS (D: DEVNAGARI; B: BANGLA)

Document Type	*Old Books*	*Photocopy*	*Newspaper*	*Fax-Document*	*Total*
# Documents	40 (D) 40 (B)	25 (D) 25 (B)	25 (D) 25 (B)	10 (D) 10 (B)	100 (D) 100 (B)
# Characters	55363 (D) 76126 (B)	38136 (D) 40582 (B)	35325 (D) 38749 (B)	3843 (D) 4123 (B)	132667 (D) 158579 (B)
# Touching characters	6201[11.2%](D) 9090[12.1%](B)	3089[8.1%](D) 4586[11.3%](B)	2014[5.7%](D) 2712[7.0%](B)	273[7.1%](D) 326[7.9%](B)	11577[8.7%](D) 16714[10.5%](B)

(a) (b)

Fig. 4. Some touching characters found in (a) Devnagari and (b) Bangla documents.

- *Observation 3:* Touching characters generally have an aspect ratio (width/height) larger than that of single isolated characters.
- *Observation 4:* A single black run is encountered at the touching position in most images.
- *Observation 5:* The vertical thickness of the black blob at the touching position is usually small compared with the thickness of other parts.
- *Observation 6:* The constituent characters have a tendency to touch each other at the middle of the middle zone rather than at the top or bottom of the zone.
- *Observation 7:* The character parts generate uncommon (quite a few in number) stroke patterns above and below the touching points.

III. FUZZY MULTIFACTORIAL ANALYSIS

In 1982, Wang [24] first defined the concept of factor spaces and applied it to the study of artificial intelligence [25]. In his factor space theory, the word *factor* is the primary term that is denoted by a noun. It has properties like *state* denoted by a numeral and *characteristic* by an adjective. The factor *weight*, for instance, is a noun; 120 lb, 150 lb, etc., are its state; heavy, light, etc., are its characteristics.

Li and Yen [26] have discussed four types of factors, namely

1) *Measurable Factors*—factors like time, height, weight, etc., that are measurable and for which the state space (i.e., the range of values) can be represented as a discrete subset of an interval. For example, state space for the factor *weight* can be represented by subset (1, 2, ..., 300).
2) *Nominal Factors*—factors that are qualitative in nature and whose state space are sets of terms. For instance, the state space of the factor *religion* is given by (Hinduism, Christianity, ..., Islam).
3) *Degree Factors*—factors for which the state spaces are usually the interval [0, 1]. Degree of similarity, feasibility, portability, etc., are the examples under this category.
4) *Switch Factors*—factors for which only two values (*yes* or *no*) are possible.

Because of their boolean characteristics, we call class 4) boolean factors.

For class 3), we call the factors *fuzzy factors* and employ them in our problem. In this connection, a brief discussion on fuzzy multifactiorial analysis is given in the following.

Consider a decision-making problem where we try to find an optimal solution or solution-set against some objectives. At first, we identify the objects based on the objectives to be satisfied. For instance, in segmenting touching characters, each pixel column in the touching-character image can be viewed as an object, and our objective is to find column(s) that segment the characters in an optimal way. Objects are related to a number of factors, and the identification of an object depends of the states of these factors. For instance, an employee in an office has his/her own data: name Mr. X, age, 30 years, sex, male, and department, accounts, etc. So, an employee is determined by specifying the states of each factor relevant to him/her. By specifying a number of factors, Mr. X can be uniquely determined and described by them. Hence, an object like a point in Cartesian space can be viewed as residing in a space constructed by the factors.

However, an object may not have significance to an arbitrary factor. For example, it is meaningless to discuss factors such as *time* or *mass* of a pixel column in segmenting touching characters. So, we say a factor f is relevant to an object o only if f has significant importance to o. By specifying the factors relevant to the objects identified under the problem specification, our goal is to uniquely determine the object(s) that gives the optimal solution.

After the objects and the factors relevant to them are identified, we go for evaluation of the factors against each object. Suppose n mutually independent factors $f_1, f_2, \ldots, f_n$ are identified as relevant to an object o and, say, m such objects $o_1, o_2, \ldots, o_m$ are located under the problem to be solved. Let $f = f_1, f_2, \ldots, f_n$ be the set of n factors. Then, for each of the m objects $o_i: (i = 1.2\ldots, m)$, we get n different values (or states) corresponding to the n factors, and in total, m different evaluations are obtained: one per object. Let $E \quad e_1, e_2, \ldots, e_m$ be the set of m evaluations for m objects, and for each e_i, there are n values $v_{i1}, v_{i2}, \ldots, v_{in}$: one for

TABLE II
NUMBER OF CONSTITUENT CHARACTERS TOUCHING EACH OTHER

Script	*#Touching characters*	*An image of touching characters consists of*		
		Two characters	Three characters	Four characters
Devnagari	11577	11183 (96.6%)	394 (3.4%)	nil
Bangla	16714	16277 (91.4%)	953 (5.7%)	484 (2.9%)

each of the n different factors. We represent these values in a matrix called the evaluation matrix V

$$V = \begin{bmatrix} v_{11} & v_{12} & \cdots & v_{1m} \\ v_{21} & v_{22} & \cdots & v_{2m} \\ \vdots & \vdots & \cdots & \vdots \\ v_{n1} & v_{n2} & \cdots & v_{nm} \end{bmatrix} \quad (1)$$

where n is the number of factors, m is the number of evaluations, and each column in V represents one evaluation.

Now, from this matrix, it is very difficult to choose any evaluation as the solution because each of them consists of n different values for n different factors. So, we design a function M_n that maps a n-dimensional vector $f = (f_1, f_2, \ldots, f_n)$ into a 1-D scalar. Mathematically, it is represented as $M_n(f) = M_n(f_1, f_2, \ldots, f_n)$. Since M_n is a function of factors, it is called a multifactorial function.

Here, we are interested in factors that are fuzzy in nature. So, the state space of the factor $(f_i)_i^{\varepsilon}{}_1$ is represented by closed unit intervals [0, 1]. We also restrict the design of M_n so that the synthesized value should never be greater than the largest of the component values and should never be less than the smallest of the component values, i.e.,

$$\bigwedge_{i=1}^{n} \leq M_n(f_1, f_2, \ldots, f_n) \leq \bigvee_{i=1}^{n} f_i. \quad (2)$$

In the literature, a function like M_n that satisfies the above condition in (2) is called an *additive standard multifactorial* (ASM) function. Next, by applying such an ASM function M_n on V, we make a multifactorial evaluation as follows:

$$V' = (v_1, v_2, \ldots, v_m) \quad (3)$$

where $v_i = M_n(v_{1i}, v_{2i}, \ldots, v_{ni})$ for $i = 1$ to m.

As we deal with fuzzy factors, all the v_{ij}s are in [0, 1]. Since M_n is an ASM function, the values of all v_is for $(i = 1, 2, \ldots, m)$ are in [0, 1]. Next, a simple search operation is performed to select v_k, which is the maximum over all v_is, i.e., $v_k = \max(v_1, v_2, \ldots, v_m)$ to indicate that the evaluation corresponding to v_k (i.e., the kth column in matrix v) should be adopted as the optimal solution. However, in our method, we try to find several v_ks to achieve the best possible segmentation for the touching characters, details of which are presented in the next section.

IV. IDENTIFICATION AND SEGMENTATION OF TOUCHING CHARACTERS

Earlier, it was mentioned that segmentation of touching characters needs some extra effort compared with the general character segmentation. So, the system should not invoke this computationally expensive method for segmenting every character. One approach is to use this technique only for the characters, which are rejected by the recognition engine. However, the recognizer also rejects some characters other than the touching ones. Hence, it is better to identify the touching characters prior to invoking the special routine for segmenting them.

A. Identification of Touching Characters

In our system, touching characters are identified by a technique based on multifactorial analysis. For this purpose, two factors are defined as follows: measure of dissimilarity (f_{md}) and aspect ratio (f_{ar}).

The first factor f_{md} is designed based on Observation 2, as listed in Section II-C. Mathematically, f_{md} is represented by $f_{md} = 1 - d_{\text{off}}/d$, where d is the minimum similarity distance for a target character against a set of stored prototypes, and d_{off} is the offset distance used by the character classifier (which is the character for which d less than d_{off} is accepted or otherwise rejected). Unlike Roman script, in Devnagari, or Bangla scripts, the combined shape of touching characters is markedly different from that of any valid character. Thus, when the classifier tries to recognize touching characters, it gives a distance measure d that is much larger than d_{off}. The factor f_{md} takes care of this phenomenon.

The second factor f_{ar} is defined as $f_{ar} = e^a/(1 + e^a)$, where $a = w/h$, w, and h are the width and height of the minimum upright rectangular bounding box of a character, respectively. As per our Observation 3 in Section II-C, it is to be noted that f_{ar} is designed to reflect the fact that touching characters generally have an aspect ratio (w/h) larger than that of single isolated characters.

Now, for each character rejected by the classifier, its f_{md} and f_{ar} are evaluated. Next, a multifactorial function M_{id} is used to map the 2-D vector into a scalar quantity as $M_{id}(f_{md}, f_{ar}) = 1/2(f_{md} + f_{ar})$. Note, that the spaces constructed by both the vectors are theoretically bounded by the interval [0, 1], and M_{id} retains the properties of an ASM function, as given in (2). Basically, M_{id} determines a degree of membership for each of the characters for which it is evaluated. If this degree of membership associated with a component is above a threshold, then it is deemed as touching character.

B. Multifactorial Analysis to Find the Cut Position

In our approach, multifactorial analysis used to select the cut columns for separating the touching characters is based on five fuzzy factors. The factors are as follows:

- f_{ic}: inverse crossing-count $= c^{-1}$, where c is the vertical crossing count (number of white to black transitions) for a pixel column.

The factor f_{ic} is designed to reflect the property stated in Observation 4 in Section II-C. Hence, if a column scan encounters crossing count larger than 1, the column is less favorable to be

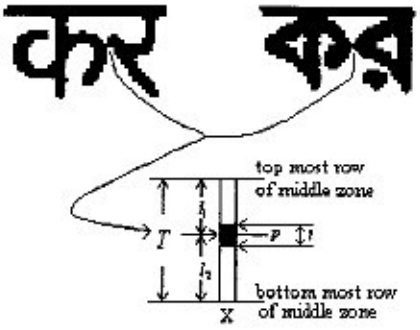


Fig. 5. Features extracted during column scan. X is a pixel column. T is the height of the middle zone. p is the midpoint of the black run whose thickness is denoted by t. l_1 and l_2 are distances of point p from the highest and lowest row of the middle zone, respectively.

a cut column. Moreover, as per our Observation 5, the vertical thickness of the black blob at the touching point is always small compared with the thickness of the other character parts; so, we design the second factor (f_{mt}) to take care of this as follows:

- f_{mt}: measure of blob thickness $= 1 - t/T$, where t (see Fig. 5) is the number of black pixels (measured in the middle zone of a character) encountered in one column scan, and T is the height of the character's middle zone.

  Here, it is to be noted that for evaluation of f_{ic} and f_{mt}, no extra computation effort is needed to compute c, t, and T. These are precomputed by the recognition engine for character classification, and hence, these values are only reused in the routine for separation of touching characters.

  To take care of Observation 6 (in Section II-C), column scans are observed. Whenever a black part is encountered in any column scan, the factor f_{dm} defined in what follows is used to check whether it belongs to the middle region of the character's middle zone.
- f_{dm}: degree of middleness $= \min(l_1, l_2)/\max(l_1, l_2)$, where l_1 and l_2 are diagrammatically explained in Fig. 5.

  In Fig. 5, p is the midpoint of the black run, and l_1 and l_2 are the distances of p from the uppermost and lowermost row of the middle zone, respectively. Note that f_{dm} becomes 1 if the black run is exactly at the middle, and it becomes less than 1 if the black run is away from the middle region.

The other two factors are designed according to our Observation 7, stated in Section II-C. To formulate them, we carry out a statistical analysis of the database of touching characters and analyze different stroke patterns formed at touching points. For this purpose, the stroke patterns above and below the touching points are examined. This is done by using a 5×5 grid, whose (3,3) element (i.e., the innermost square) is considered as the touching point. Five different stroke patterns are detected above the touching points. We call these patterns u patterns. Similarly, five different patterns (l-pattern) are detected below the touching points. These patterns are shown in Fig. 6 with example image of touching characters.

In the fuzzy frequency analysis, these patterns are treated as fixed elements $x_0 \in X$ (universe) and the database of sample touching characters A_* (say, N number of samples are there) is used to compute the relative frequency of x_0 in A_*. This is computed as follows. The relative frequency of x_0 in $A_* =$ (number of observations of $x_0 \in A_*) \div (N)$. As N increases, the relative frequency tends to stabilize or converge to a fixed number; this number is called the *degree of membership* of x_0 in A_*. Table III shows the *degree of membership* for the patterns listed in Fig. 6.

In each column scan, presense of any u pattern and l pattern are recorded in two boolean variables B_u and B_l, respectively, and based on their values, a column gets two different predefined weights, W_u and W_l, for u pattern and l pattern, respectively. These weights are computed as follows. If B_u, then $W_u = w_u$, else $W_u = 1 - w_u$, and similarly, if B_l, then $W_l = w_l$, else $W_l = 1 - w_l$. In our implementation, w_u and w_l both are less than 1 but greater than 0.5.

The fourth factor f_{up} is evaluated by the following equation, which combines weight W_u and the membership functions $\dot{A}_u$ (= membership of P_{ui}, as given in Table III).

- f_{up} $W_u + |(1 - w_u) \times \dot{A}_u| \wedge B_u$. In the above equation, the symbol ($\wedge$) represents logical AND operation, i.e., the term $[(1 - w_u) \times \dot{A}_u]$ is added with W_u only if B_u is true. Similarly, the fifth factor f_{low} is evaluated by the following equation:
- f_{low} $W_l + |(1 - w_l) \times \dot{A}_l| \wedge B_l$, where $\dot{A}_l$ is the membership function whose values are associated with ($= P_{li}$) in Table III.

Next, let m be the total number of pixel columns in any image of touching characters. For all the m columns, the above five factors (f_{ic}, f_{mt}, f_{dm}, f_{up}, and f_{low}) are evaluated, and a $5 \times m$ one-factor evaluation matrix is formed as follows:

$$V_s = \begin{bmatrix} f_{ic}^1 & f_{ic}^2 & \cdots & f_{ic}^m \\ f_{mt}^1 & f_{mt}^2 & \cdots & f_{mt}^m \\ f_{dm}^1 & f_{dm}^2 & \cdots & f_{dm}^m \\ f_{up}^1 & f_{up}^2 & \cdots & f_{up}^m \\ f_{low}^1 & f_{low}^2 & \cdots & f_{low}^m \end{bmatrix}. \quad (4)$$

Each column in matrix V_s represents each pixel column in the image and consists of a 5-D vector that reflects five different aspects for that column to be a cut column. Next, these five aspects get combined and mapped into a 1-D scalar by an ASM function M_s. The function M_s transforms the $5 \times m$ matrix V_s into a $1 \times m$ matrix V_s' as follows:

$$V_s' = (f_s^1, f_s^2, \ldots, f_s^m) \quad (5)$$

where f_s^i $M_s(f_{ic}^i, f_{mt}^i, f_{dm}^i, f_{up}^i, f_{low}^i)$ $1/5(f_{ic}^i + f_{mt}^i + f_{dm}^i - f_{up}^i + f_{low}^i)$; i varies from 1 to m.

Since the state spaces of all the five factors are theoretically bounded by the interval [0, 1] and M_s retains the property of an ASM function (discussed in Section III), the state space for f_s is also bounded by the interval [0, 1]. Actually, M_s is used to give each pixel column (i) a degree of membership f_s^i that reflects the possibility of the ith column to be a cut column for separating characters. Higher f_s value indicates larger possibility for that column to be a cut column.

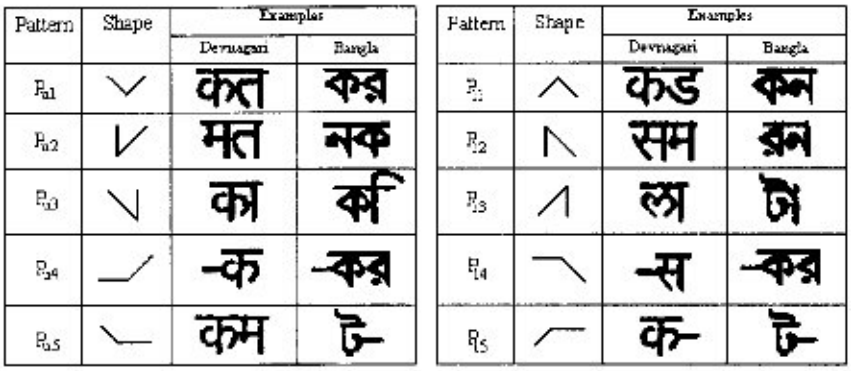


Fig. 6. Different shapes formed at touching points. (a) P_{ui} are the u patterns that show shapes above a touching point. (b) P_{li} are l patterns that show shapes below a touching point.

TABLE III
DEGREE OF MEMBERSHIP FOR THE U PATTERNS AND L PATTERNS

u-patterns	*count* (total:28291)	*Membership* (frequency)	*l-patterns*	*count* (total:28291)	*Membership* (frequency)
P_{u1}	15305	0.541	P_{l1}	17682	0.625
P_{u2}	8402	0.297	P_{l2}	5658	0.2
P_{u3}	2292	0.081	P_{l3}	2122	0.075
P_{u4}	1528	0.054	P_{l4}	2122	0.075
P_{u5}	764	0.027	P_{l5}	707	0.025

```
Algorithm I. Confirmation of cut columns
confirmCut(P, pass)
Comment: P[1, m] is the image of the
pattern to be segmented. Variable pass
is used to restrict the number of al-
ternatives tried in each pattern. The
variable m is the width[P] in terms of
pixel-column. The function parent[P] re-
turns the parent image of P and it re-
turns NULL if P is the initial pattern.
BEGIN
Step1: if(P == NULL)
  error msg ("Segmentation Failure."); re-
turn;
Step2: m = width[P];
Step3: if(pass == 1){
  Find i for which f_s^i = max(f_s^1, f_s^2, ..., f_s^m);
  //f_s^i values are given by (5).
  P1  P[1, i] and P2  P[i + 1, m];
 }
 else{
  Find j for which f_s^j is the second
highest
  among all f_s^i where i  1, 2, ..., m;
  P1 = P[1, j]  and P2 = P[j  1, m];
 }
Step4: Call Character Classifier to recog-
nize both P1 and P2.
Step5: if both P1 and P2 are recognized
then i is the correct cut column. Return;
Step6: if P1 is recognized and P2 is not
recognized then confirmCut(P2, 1); // Re-
cursive call
Step7: if P1 is not recognized and P2 is
recognized then confirmCut(P1, 1); // Re-
cursive call
Step8: if both P1 and P2 are not recog-
nized then
   if (pass  1)
    pass = 2; GOTO step3.
   else
    confirmCut(parent[P], 2); //Recursive
call
END
```

C. Confirmation of the Cut Columns

Researchers proposed various alternative approaches for the confirmation of cut columns. Casey and Nagy [1] proposed recursive tests of all combinations of admissible separation boundaries until they either exhaust the set of cut columns or find an acceptable segmentation. Liang *et al.* [9] developed another algorithm that implements a forward segmentation or a backward merge procedure based on the output of a character classifier. Both procedures are slow since the character classifier spends time in attempting to recognize many patterns that are not valid. Tsujimoto and Asada [2] constructed a decision tree and a set of additional rules to obtain the correct character component sequences. Their algorithm also requires intensive computation to build a decision tree and search for a correct path.

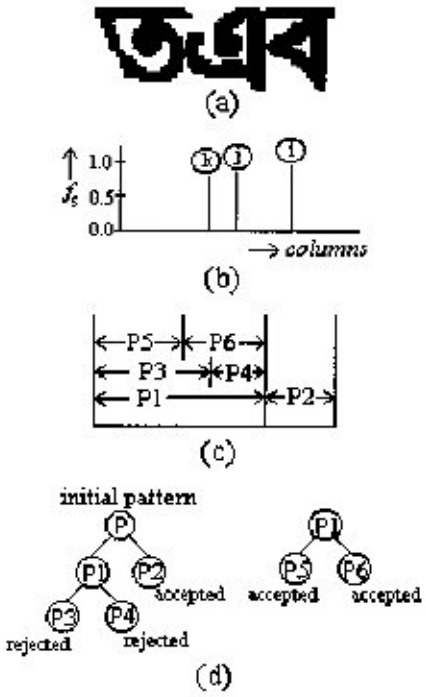


Fig. 7. Confirmation of cut columns for a Bangla triplet.

In our approach, we have attempted to reduce the computation by predicting the most favorable cut column before character classifier is used to confirm it. This concept is borrowed from the predictive parser concept that is well known in the field of compiler design for the programming languages [19]. Here, the column having the highest f_s value is predicted as the most favorable cut column. The components formed by this cut column are sent to the classifier, and based on its decision, the algorithm goes forward to the next cut column or tries with the column having the second best f_s value. The algorithm terminates when all segments generated by selected cut columns pass through the character classifier. The algorithm in a pseudo code is given in Algorithm I.

To reduce computation time, for each pattern, only the top two f_s values that belong to the current pattern are used as two successive predicted cut columns. The test results (presented in the next section) show remarkable accuracy in finding the correct cut columns. This fact attests to our approach of using f_s values to make the right predictions.

Fig. 7 illustrates the algorithm for a touching triplet in Bangla script. Fig. 7(b) shows the three cut columns (columns) having top three f_s values for the touching character shown in Fig. 7(a). Column i has the highest f_s value, so it is predicted as the most favorable cut column, yielding patterns P1 and P2. The classifier recognizes P2 as a valid character, whereas P1 is rejected [Fig. 7(d)]. P1 is then segmented by the cut column (column j) having the highest f_s value within P1 [see Fig. 7(b) and (c)]. This cut yields P3 and P4, both of which are rejected by the classifier [Fig. 7(d)]. So, the algorithm chooses the next cut column (column k having the second highest f_s value within P1), yielding patterns P5 and P6, both of which are accepted by the character classifier. At this stage, the algorithm terminates for this touching character.

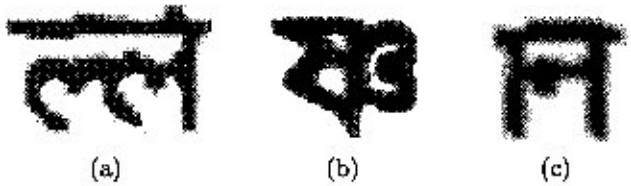


Fig. 8. Examples show identification error.

V. Test Results

In our experiment, we have used documents listed in Table I. The documents are scanned at various resolutions using a flatbed scanner. All algorithms in this paper are written in C and executed on a Pentium-II 333–MHz machine.

A. Accuracy in Identifying Touching Characters

Our proposed approach has two parts in dealing with touching characters: i) identification and ii) segmentation of touching characters. Each of these two modules is tested in isolation so that their performance can be quantified separately. The experiment is done on 11 577 Devnagari touching characters (16 714 for Bangla), as listed in Table II. Test results show an overall 98.87% (for Devnagari) and 98.63% (for Bangla) accuracy in identifying the touching characters. Two types of errors occur during identification.

i) *False identification:* some valid characters (not touching but rejected by the initial classifier) are identified as touching characters.
ii) *Rejection:* some touching characters are not identified at all.

The first type of errors occur in the case of some compound characters (where two basic characters combine each other side by side) present in both the scripts, i.e., when the character classifier could not recognize some compound characters (because of shape deformation, presence of noise, or scanning resolution, etc.). These rejected characters, in some cases, give high value for f_{ov}, leading to error in the identification module. However, the rate for such error is as low as 0.46% (for Devnagari) and 0.58% (for Bangla). Fig. 8(a) and (b) shows two examples of such compound characters.

We find two reasons for the second type of errors: i) wide variation in character widths and ii) high degree of degradation in some scanned documents. In rare occasions, the width of a touching-character pattern (where one or more constituent characters are relatively thin in width) is even less than that of a single basic character. In such cases, f_{ov} is low, leading to error in the identification module. In another situation, the original classifier may give high similarity measure (but rejects the character since the measure is less than predefined threshold) for a deformed touching character, and f_{md} gets low value. Fig. 8(c) shows an example, which was rejected by the character classifier but not identified as a touching pattern.

B. Segmentation Accuracy

The touching character segmentation module shows 98.92% and 98.47% accuracy for Devnagari and Bangla scripts, respec-

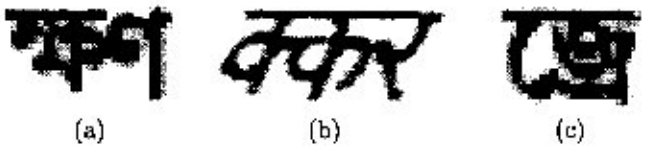


Fig. 9. Some examples show segmentation failure.

tively. Three different sources are identified behind the errors occurring at this stage:

i) constituent characters that deeply touch each other, and the touching point is as thick as (or even thicker than) the average character stroke [see Fig. 9(a)]; the segmentation algorithm fails for this reason in 0.73% and 0.98% of the cases for Devnagari and Bangla, respectively;
ii) italic style of characters [see Fig. 9(b)]; a phenomenon causing segmentation error in 0.23% and 0.34% of the cases for Devnagari and Bangla, respectively;
iii) unpredicted touching position [see Fig. 9(c)] due to excessive degradation of the document.

For this reason, we encounter segmentation failure in 0.12% and 0.21% of the cases for Devnagari and Bangla, respectively.

Segmentation errors can be reduced by allowing more than two predicted cut columns, but since it slows the segmentation speed, we restrict the number of predictions to two. On the other hand, use of a language bigram model will hopefully improve the overall segmentation score. In this technique, multiple cut columns are selected based on classifier's positive response, and then, final acceptance of a cut column is decided by looking at the bi-gram or tri-gram statistics of the language.

C. Effect of Scanning Resolution

To test how sensitive our technique is to the variation of scanning resolutions, we selected 40 representative documents (20 for each script) from our dataset (see Table I). Each of these documents were scanned at five diffrent resolutions: 75, 150, 200, 250, and 300 dpi. Character font size (in points) in these documents varies from 12 to 24. It is observed that there is a negligible change in number of touching characters in images scanned at 300 and 250 dpi, but below 250 dpi, the number of touching characters gradually increases (for both Devnagari and Bangla documents) with the decrese in resolution. On an average, if x is the number of characters in a document scanned at 250 dpi (or higher), then gradual increments in number of touching characters for that document at various lower resolutions are as follows:

200 dpi	150 dpi	75 dpi
$0.027x$	$0.052x$	$0.089x$

Though the number of touching characters increases with the decrease in resolution, our algorithms show stable performance both for identifying and segmenting touching characters. In fact, this consistency in performance shows the adequacy of our feature selection process. It is examined that accuracy in identifying touching characters is quite high as 99.11% (Devnagari) and 98.95% (Bangla) when documents are scanned at 250 dpi or more and never less than 97.65% (Devnagari) and 97.20% (Bangla), even when scanning resolution is as low as 75 dpi. Similarly, segmentation accuracy is 99.30% (Devnagari) and 98.90% (Bangla) for documents scanned at 250 dpi or more, and these figures for 75 dpi are 97.85% (Devnagari) and 97.30% (Bangla).

D. Improvement of OCR Accuracy

The addition of an extra module for processing of touching characters empowers our original OCR system [16] to process degraded documents having large number of touching characters with higher recognition accuracy. This achievement has been tested by using documents of types listed in Table I. Experimental results show remarkable improvement in the OCR recognition rate for handling these types of documents. Fig. 10(b) shows our original OCR output for an input photocopied document shown in Fig. 10(a) (Hindi), and Fig. 10(c) shows the result on the same input document from the same OCR integrated with the module that takes care of the touching characters. Fig. 11 presents such comparative studies for ten documents (details listed in Table IV) for both types of scripts.

E. System Throughput and Efficiency Measure

Sometimes, addition of a new module to an existing system may decrease the overall throughput (i.e., the amount of work that can be performed by a computer system or component in a given period of time [27]) of the system as it does some extra computations. Hence, effort should be made so that throughput never suffers drastically. Keeping this in mind, we have attempted to design the module for processing touching characters in a way that the overall OCR throughput never goes down below an acceptable level. We have checked this by defining throughput for our OCR as $T = O/t$, where

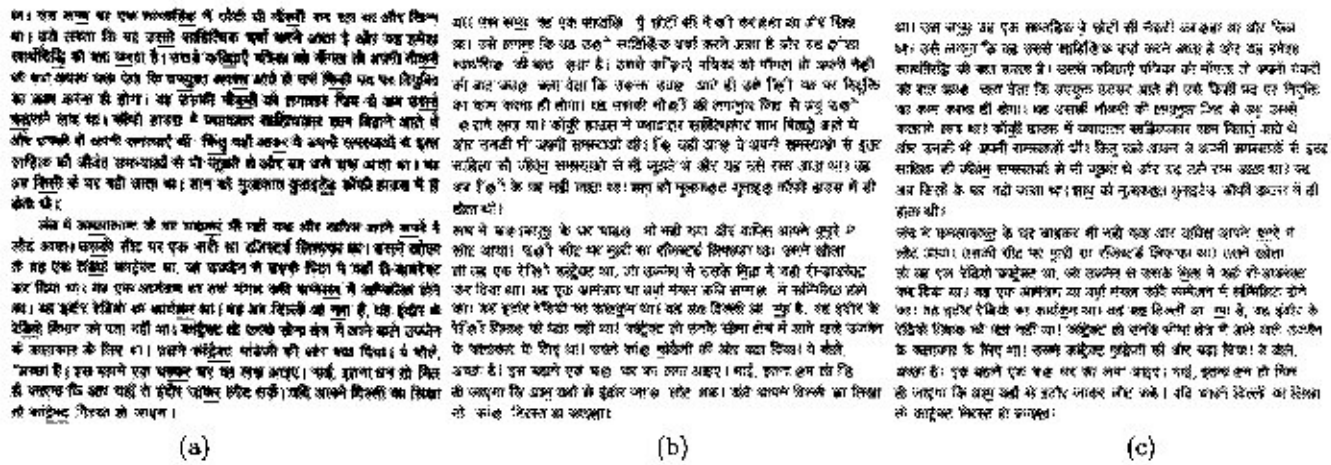
T system throughput;
O total number of characters properly recognized by the OCR;
t total time elapsed for this operation.

The term t includes time for processing all the OCR steps, starting from gray-to-binary conversion of a digitized document till production of the final output.

A test on the dataset shown in Table I shows that in the modified system, the overall time elapsed for processing a document is slightly more than the time taken by our original system, but the throughput never degrades. On the other hand, for the documents having a large number of touching characters, there is a considerable improvement in the system throughput. The results for ten representative documents (five for each script) are illustrated in Table IV.

An efficiency measure of the algorithm is important because each time a cut column is selected, the character classifier is called to recognize the segmented components generated by the cut column. So, the more the number of invalid cut columns, the more intensive the overall computation. Let N_v be the number of valid cut columns and N_t be the total number of cut columns checked to find valid cuts that we define efficiency $= (N_v * 100)/N_t$.

In our method, we select cut columns by looking at the prediction generated by Algorithm I. This technique generates a number of unsuccessful cut columns less than the number that would have been generated if methods (that select cut columns



(a) (b) (c)

Fig. 10. Improvement of OCR accuracy. (a) Input Hindi photocopied document contains 835 characters out of which 66 (7.9%) are touching shown by underline. (b) Original OCR (without any module to process touching characters) output and (c) modified OCR (integrated with the module that takes care of touching characters) output. Output in (b) and (c) show recognition accuracy of 89.34% and 96.89%, respectively. Underlined characters are misclassified , and characters rejected by the OCR are represented by the symbol "@."

TABLE IV
EVALUATION OF SYSTEM THROUGHPUT (DEV: DEVNAGARI DOCUMENTS; BAN: BANGLA DOCUMENTS)

Documents	*Total no. of char.*	*Total no. of touching char.*	*No of char. properly recognized by*		*Time taken by (in seconds)*		*Throughput (char./sec)*	
			Original OCR	Modified OCR	Original OCR	Modified OCR	Original OCR	Modified OCR
DEV01.tif	603	7	591	597	14	15	42	40
DEV02.tif	813	46	756	801	17	20	44	40
DEV03.tif	742	61	671	729	15	18	45	41
DEV04.tif	1305	179	1112	1289	26	30	43	43
DEV05.tif	1496	289	1183	1478	26	31	46	48
BAN01.tif	1132	17	1103	1118	24	25	46	45
BAN02.tif	1167	77	1076	1149	25	28	43	41
BAN03.tif	1662	139	1497	1634	32	35	47	47
BAN04.tif	1960	218	1714	1928	36	40	48	48
BAN05.tif	2625	432	2151	2579	45	50	48	52
Total	13505	1465	11854	13302	260	292	46	46

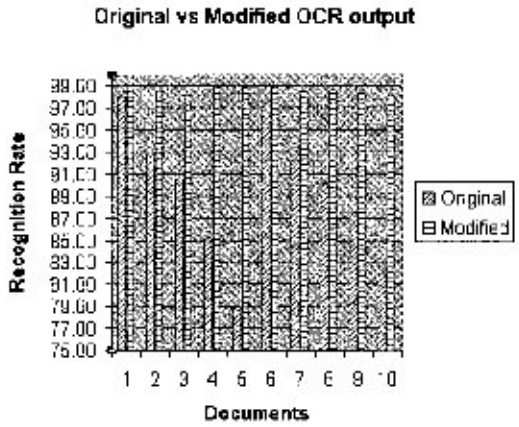


Fig. 11. Improvement of OCR accuracy. The first five documents are in Devnagari, and the next five are in Bangla. Documents in each script are in ascending order on the number of touching characters present in them.

from left to right direction) described in [1] and [9] were followed. Experimental results show that our proposed technique has efficiency of 79.64 and 78.99% for Devnagari and Bangla, respectively. This indicates that out of 100 cut columns tested by our technique, more that 79 cut columns (78 for Bangla) are valid.

On the other hand, if we use some threshold on f_A^* values [see (5)] rather than associating the degree of membership and then select cut columns from left to right, as discussed in [1] and [9], we get an efficiency of 57.48 and 53.65% for Devnagari and Bangla, respectively. This shows that our idea of using fuzzy multifactorial analysis is quite justified.

VI. CONCLUSION

We have developed an effective strategy for segmenting touching characters that appear in Devnagari and Bangla documents. To the best of our knowledge, this is the first study of its kind of these two major Indian scripts. Our technique shows high accuracy with a reasonable computational effort. It seems that this technique can be suitably modified for segmenting touching characters in other scripts like Roman (English), etc. Another important aspect of this study is the application fuzzy multifactorial analysis in a practical problem involving document image analysis. It is likely that this approach will find applications in other document analysis problems.

ACKNOWLEDGMENT

The authors sincerely thank the reviewers of the conference as well as those of this journal for their comments and suggestions in improving the quality of the manuscript.

REFERENCES

[1] R. G. Casey and G. Nagy, "Recursive segmentation and classification of composite character patterns," in *Proc. 6th Int. Conf. Pattern Recognition*, Munich, Germany, 1982, pp. 1023–1026.

[2] S. Tsujimoto and H. Asada, "Major components of a complete text reading system," *Proc. IEEE*, vol. 80, pp. 1133–1149, July 1992.

[3] H. Fujisawa, Y. Nakano, and K. Kurino, "Segmentation methods for character recognition: From segmentation to document structure analysis," *Proc. IEEE*, vol. 80, pp. 1079–1092, July 1992.

[4] M. Bokser, "Omnidocument technologies," *Proc. IEEE*, vol. 80, pp. 1066–1078, July 1992.

[5] T. Nartker, "ISRI Annu. Rep.," Univ. Nevada, Las Vegas, 1993.

[6] D. G. Elliman and I. T. Lancaster, "A review of segmentation and contextual analysis techniques for text recognition," *Pattern Recognit.*, vol. 23, no. 3/4, pp. 337–346, 1990.

[7] R. G. Casey and E. Lecolinet, "A survey of methods and strategies in character segmentation," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 18, July 1996.

[8] V. A. Kovalevsky, *Character Readers and Pattern Recognition*. Washington, DC: Spartan, 1968.

[9] S. Liang, M. Shridhar, and M. Ahmadi, "Segmentation of touching characters in printed document recognition," *Pattern Recognit.*, vol. 27, no. 6, pp. 825–840, 1994.

[10] T. Bayer and U. Kressel, "Cut classification for segmentation," in *Proc. Int. Conf. Document Anal. Recognit.*, Tsukuba Science City, Japan, 1993, pp. 565–568.

[11] S. Kahan, T. Pavlidis, and H. S. Baird, "On the recognition of printed characters of any font and size," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 9, no. 2, pp. 274–288, 1987.

[12] Y. Lu, "On the segmentation of touching characters," in *Proc. Int. Conf. Document Anal. Recognit.*, Tsukuba Science City, Japan, 1993, pp. 440–443.

[13] J. Rocha and T. Pavlidis, "A shape analysis model with applications to a character recognition system," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 16, pp. 393–404, Apr. 1994.

[14] B. B. Chaudhuri and U. Pal, "A complete printed Bangla OCR system," *Pattern Recognit.*, vol. 31, pp. 531–549, 1998.

[15] ——, "An OCR system to read two Indian language scripts: Bangla and Devnagari (Hindi)," in *Proc. 4th Int. Conf. Document Analysis Recognition*, Ulm, Germany, 1997, pp. 1011–1016.

[16] B. B. Chaudhuri, U. Garain, and M. Mitra, "On OCR of the most popular Indian scripts: Devnagiri and Bangla," in Visual Text Recognition and Document Processing , N. Murshed, Ed. Singapore: World Scientific .

[17] U. Garain and B. B. Chaudhuri, "On recognition of touching characters in printed Bangla documents," in *Proc. Indian Conf. Computer Vision, Graphics, Image Processing*, S. Chaudhury and S. K. Nayar, Eds., Delhi, India, 1998, pp. 377–380.

[18] H.-X. Li, "Multifactorial functions in fuzzy sets theory," *Fuzzy Sets Syst.*, vol. 35, no. 1, pp. 69–84, 1990.

[19] A.V. Aho, R. Sethi, and J.D. Ullman, *Compilers: Principles, Techniques and Tools*. Reading, MA: Addison-Wesley, 1986.

[20] A. Hashizume, P. S. Yeh, and A. Rosenfeld, "A method of detecting the orientation of aligned components," *Pattern Recognit. Lett.*, vol. 4, pp. 125–132, 1986.

[21] L. O'Gorman, "The document spectrum for page layout analysis," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 15, pp. 1162–1173, Nov. 1993.

[22] H. Yan, "Skew corrections of document images using interline cross-correlation," in *Comput. Vision, Graphics, Image Process.*, vol. 55, 1993, pp. 538–543.

[23] B. B. Chaudhuri and U. Pal, "Skew angle detection of digitized Indian script documents," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 19, pp. 182–186, Feb. 1997.

[24] P.-Z. Wang and M. Sugeno, "The factor fields and background structure for fuzzy subsets," *Fuzzy Math.*, vol. 2, no. 2, pp. 45–54, 1982.

[25] P.-Z. Wang, "A factor space approach to knowledge representation," *Fuzzy Sets Syst.*, vol. 36, pp. 113–124, 1990.

[26] H. X. Li and V. C. Yen, *Fuzzy Sets and Fuzzy Decision-Making*. Boca Raton, FL: CRC, 1995.

[27] "Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries," IEEE, New York, 1990.

Utpal Garain was born in Suri, W.B., India, in 1972. He received the B.E. and M.E. degrees in computer science and engineering from Jadavpur University, Kolkata, India, in 1994 and 1997, respectively.

He started his career as a software professional in industry, and later on, joined the Indian Statistical Institute, Kolkata, as a research personnel, where he is now full-time faculty. He is one of the key scientists involved in the development of a bilingual (Devnagari and Bangla) OCR system, which is first of its kind in India. In the last three years, he has published several technical papers in reputed international journals and conferences. His areas of interest include digital document processing including OCR for Indian language scripts, document data compression, etc.

Bidyut B. Chaudhuri (F'01) received the B.Tech. degree and the M.Tech degree from Calcutta University, Kolkata, India in 1972 and 1974, respectively. He received his Ph.D. in 1980 from the Indian Institute of Technology, IIT, Kanpur.

He is currently the Head of the Computer Vision and Pattern Recognition Unit, Indian Statistical Institute, Kolkata. His research interests include pattern recognition, image processing, computer vision, natural language processing (NLP), and digital document processing including optical character recognition. He has published more than 200 research papers in reputed international journals and has written three books. He is serving as Associate Editor of *Pattern Recognition, Pattern Recognition Letters*, and *VIVEK*, as well as guest editor of a special issue of the *Journal IETE on Fuzzy Systems*.

Prof. Chaudhuri has received many awards and prizes, including the Sir J. C. Bose Memorial Award in 1986, the M. N. Saha Memorial Award in 1989 and 1991, the Homi Bhabha Fellowship award in 1992, the Dr. Vikram Sarabhai Research Award in 1995, and the C. Achuta Menon Prize in 1996 for his contribution in the field of engineering sciences, Indian language processing, computer applications, etc. He is a fellow of IAPR, IETE (India), the National Academy of Sciences, and the National Academy of Engineering (India).