Multioriented and Curved Text Lines Extraction From Indian Documents

U. Pal and Partha Pratim Roy

Abstract—**There are printed artistic documents where text lines of a single page may not be parallel to each other. These text lines may have different orientations or the text lines may be curved shapes. For the optical character recognition (OCR) of these documents, we need to extract such lines properly. In this paper, we propose a novel scheme, mainly based on the concept of *water reservoir analogy*, to extract individual text lines from printed Indian documents containing multioriented and/or curve text lines. A reservoir is a metaphor to illustrate the cavity region of a character where water can be stored. In the proposed scheme, at first, connected components are labeled and identified either as isolated or touching. Next, each touching component is classified either straight type (S-type) or curve type (C-type), depending on the *reservoir base-area* and *envelope points* of the component. Based on the type (S-type or C-type) of a component two *candidate points* are computed from each touching component. Finally, *candidate regions* (neighborhoods of the candidate points) of the candidate points of each component are detected and after analyzing these candidate regions, components are grouped to get individual text lines.**

Index Terms—**Artistic document recognition, Bangla and Devnagari scripts, document analysis, Indian documents, text line extraction.**

I. Introduction

In OPTICAL character recognition (OCR), the text lines in a document must be located before recognition. This task involves correction of document skew, separation of text and graphics, and extraction of text lines [4], [13], [17]. When the text lines in a document image are parallel to one another (single oriented document), simple techniques like projection profile [7], [12], [18], Hough transform [11], component nearest neighbor clustering method [13] etc., are good enough to extract them. There are many artistic documents, such as advertisement, poster, or graphic illustration where text lines are not single oriented. These text lines may be multioriented or may be curved in shape. Examples of such documents are shown in Fig. 1. Extraction of individual text lines in these documents is difficult. Simple approaches like projection profile, Hough transform, or component nearest neighbor clustering will not work in these documents.

Pieces of published work on text line extraction from documents containing multioriented or curved lines are few. Goto and Asu [5] propose a local linearity based technique to identify multioriented or curved text lines from English and Chinese documents. They, at first, split the document image into some small subregions of constant width. Next, local orientation is estimated in each of these subregions. Finally, they try to extract text lines by extending local orientations of the subregions. Their proposed method cannot handle text lines of variable size characters, which is the main limitation of the method.

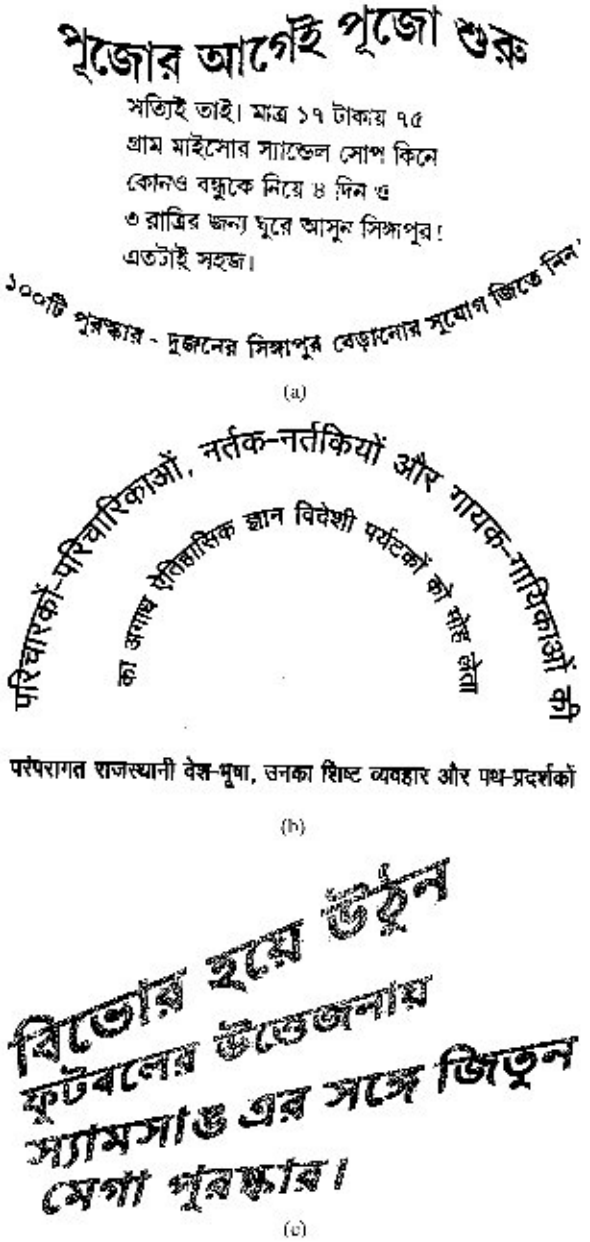

Fig. 1. Examples of document images containing multioriented and curved text lines. (a) Bangla magazine image. (b) Devnagari synthetic image. (c) Bangla newspaper image.

Manuscript received June 17, 2003; revised March 11, 2004. This paper was recommended by Associate Editor V. Govindaraju.

U. Pal is with the Computer Vision and Pattern Recognition Unit, Indian Statistical Institute, Kolkata-108, India (e-mail: umapada@isical.ac.in).

P. P. Roy is with the Tata Consultancy Service, Kolkata-91, India (e-mail: parthapratim_roy@tcscal.co.in).

Digital Object Identifier 10.1109/TSMCB.2004.827613

Fletcher and Kasturi [4] propose a method to extract text lines in arbitrary orientations from mixed text and graphics regions of English documents. The character bounding box and the Hough transform are used in the method. The proposed method cannot handle curved text lines and this method too cannot handle text lines of arbitrary size. Yan [19] proposes a fuzzy curve-tracing algorithm to detect curve text lines from English and Chinese documents. Here, at first, character pixels are grouped based on the fuzzy c-means algorithm. Each cluster center represents all its associated pixels in a class to reduce the amount of data substantially. Then, analyzing the spatial relationship, cluster centers are connected to generate the initial curve representing the text path. Finally, the text pixels are clustered again under the constraint that the path passing through the cluster centers must be smooth. Main drawback of the method is to define the smoothness term properly. Hones and Litcher [8] also propose a method to extract straight text lines in arbitrary orientations. In their method, line anchors are first found in the document image and then text lines are generated expanding the line anchors. Similar strategy has been used by Deforges and Barbara [3]. Kise *et al.* [9] propose a computational geometric approach to extract text lines from a document. Although this method can handle document images with complex layout, it cannot handle curved text lines.

All the above methods are used for non-Indian languages. Also, it can be noted that except methods used by Goto and Asu [5], and Yan [19] the other above methods cannot handle curved text lines. Although these two methods can handle straight and curve text lines in English and Chinese documents, it will not work for Indian documents containing Bangla and Devnagari scripts. This is because the characters in a word of Bangla and Devnagari scripts get connected through *head-lines* (described latter) and such behavior is absent in English and Chinese text. In this paper, we propose a novel scheme to extract multioriented and curved shape lines from documents containing Bangla and Devnagari, the two most popular Indian scripts. There is no published work in this area on Indian languages. Only one report is published to extract straight text lines from Indian documents containing Bangla and Devnagari scripts [16]. The technique described in [16] can extract only multioriented straight text lines but it cannot handle curve text lines.

In Bangla and Devnagari it is noted that most of the characters have a horizontal line at the upper part. Horizontal line of two Bangla characters is shown in Fig. 2(a). When two or more characters sit side by side to form a word, these horizontal lines touch and generate a long line called *head-line* [see Fig. 2(b)]. Here, our idea is to extract the head-line of a text component properly and to extend the head-line both ways to cluster the component with its neighbor components for individual line extraction. Head-line extraction from arbitrary oriented straight or curve text line is not simple. Here, we use a scheme based on features obtained from a concept of *water reservoir analogy* [14]. A reservoir is a metaphor to illustrate the cavity region of a character where water can be stored. Reservoir is obtained by considering accumulation of water poured from a side of a component. When two or more characters form a word in Bangla or Devnagari they create big regions (spaces) because of the touching through head-line. For example see Fig. 2(b) where big regions

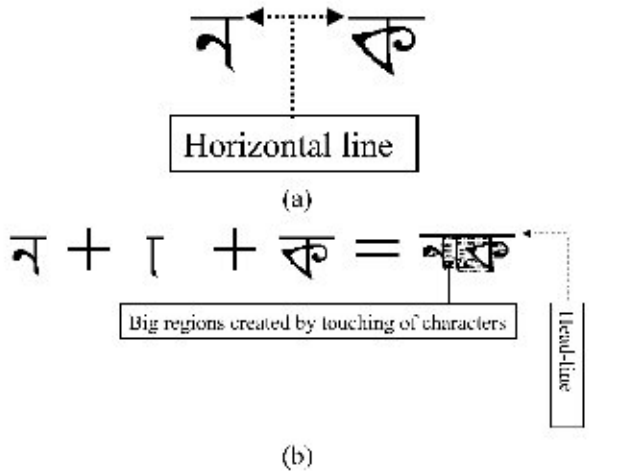

Fig. 2. (a) Horizontal lines of two Bangla characters are shown. (b) Head-line and big regions of a touching component are shown. Big regions created by touching are marked by dots.

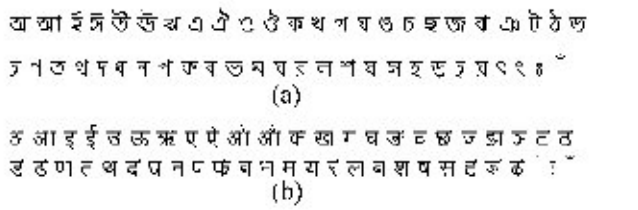

Fig. 3. Basic characters of (a) Bangla and (b) Devnagari alphabet are shown. The first 11 are vowels and the rest are consonants in both the alphabets.

formed by the characters in a word are shown. These regions are considered as reservoirs. Details about the reservoir are given in Section III.

The line extraction scheme proposed in this paper is as follows. At first, component labeling is done to get individual components. Analyzing the reservoirs obtained in a component, we compute mode (portrait, landscape, reverse portrait, reverse landscape) of the component. Next, based on mode and water reservoir features, components are classified in one of the two classes: isolated and touching. In most of the cases of Bangla and Devnagari scripts, a touching component generally represents a word because characters in a word are connected by head-line. Depending on the mode of a touching component its *envelope points* are computed. Next, based on water reservoir *base-area points* and the envelope points each touching component is classified either straight type (S-type) or curve type (C-type). Depending on the component type, two candidate points are computed from each touching component. Finally, candidate regions of these candidate points are detected and after analyzing these candidate regions, components are grouped to get individual text lines.

II. Properties of Bangla and Devnagari Script

The alphabet of the modern Bangla script consists of 11 vowels and 39 consonants. These characters may be called *basic characters*. Out of the 49 basic characters of Devnagari script, 11 are vowels and 38 are consonants. The basic characters of Bangla and Devnagari scripts are shown in Fig. 3. Writing style in both the scripts is from left to right. The concept of

Bangla vowels	আ	ই	ঈ	উ	ঊ	ঋ	এ	ঐ	ও	ঔ
Modified shape	া	ি	ী	ু	ূ	ৃ	ে	ৈ	ো	ৌ
When attached to consonant ক	কা	কি	কী	কু	কূ	কৃ	কে	কৈ	কো	কৌ
Devnagari vowels	आ	इ	ई	उ	ऊ	ऋ	ए	ऐ	ओ	औ
Modified shape	ा	ि	ी	ु	ू	ृ	े	ै	ो	ौ
When attached to consonant क	का	कि	की	कु	कू	कृ	के	कै	को	कौ

Fig. 4. Examples of Bangla and Devnagari modified characters.

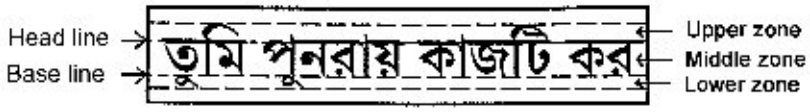

Fig. 5. Different zones of a Bangla text line.

upper/lower case is absent in Bangla and Devnagari scripts. From Fig. 3 it can be seen that most of the characters in Bangla and Devnagari scripts have a horizontal line at the upper part. From a statistical analysis, we notice that the probability that a Bangla word will have horizontal line is 0.994 and the corresponding probability for Devnagari is 0.997 [1]. Due to the high probability of head-lines in Bangla and Devnagari scripts, the use of water reservoir obtained by character touching through head-line is justified for text line extraction in these scripts.

In both Bangla and Devnagari scripts a vowel following a consonant takes a modified shape. Depending on the vowel, its modified shape is placed at the left, right (or both), or bottom of the consonant. These modified shapes are called *modified characters*. Examples of modified character are shown in Fig. 4. A consonant or vowel following a consonant sometimes takes a compound orthographic shape which we call as *compound character*. Compound characters can be combinations of two consonants, as well as a consonant and a vowel. Compounding of three or four characters also exists in these two scripts. In both the script forms, there are about 280 compound characters [2].

A Bangla or Devnagari text line can be partitioned into three zones. The *upper-zone* denotes the portion above the head-line, the *middle zone* covers the portion between head-line and base-line, and the *lower-zone* is the portion below base-line. Different zones in a Bangla text line are shown in Fig. 5. The big reservoirs are generally created in the middle zone of a text line.

Languages like Hindi, Nepali, Sanskrit, and Marathi are popularly written in Devnagari while Bangla, Assamese, and Manipuri languages are written in Bangla script. Moreover, Hindi and Bangla are the national languages of India and Bangladesh, respectively. Also, Hindi is the thirdmost and Bangla is the fifth-most popular language in the world [2].

III. Water Reservoir Principle

The water reservoir principle is as follows. If water is poured from a side of a component, the cavity regions of the component where water will be stored are considered as reservoirs. As mentioned earlier, characters in a word are connected through head-line in Bangla and Devnagari scripts. This connection generates large reservoirs which are used for line extraction purpose.

Now, we will discuss the water reservoir terms that will be used in line extraction scheme.

Top reservoir: By top reservoirs of a component, we mean the reservoirs obtained when water is poured from top of the component;

Bottom reservoir: By bottom reservoirs of a component, we mean the reservoirs obtained when water is poured from bottom of the component. A bottom reservoir of a component is visualized as a top reservoir when water will be poured from top after rotating the component by 180°;

Left (right) reservoir: If water is poured from the left (right) side of a component, the cavity regions of the component where water will be stored are considered as left (right) reservoirs.

For illustrations, see Fig. 6. In this figure, the top, bottom, left, and right reservoirs are shown in four different components. From the figure it can be noted that bigger reservoirs are obtained for portrait, reverse portrait, landscape, and reverse landscape components when water is poured from the bottom, top, left, and right side, respectively. The computation of water reservoir is simple. We find the regions of white space (pixels) in the bounding box of the component, where water could be stored. These regions are considered as the water reservoirs.

Water reservoir area: By area of a reservoir we mean the area of the cavity region where water can be stored if water is poured from a particular side of the component. The number of pixels inside a reservoir is computed and this number is considered as the area of the reservoir.

Water flow level: The level from which water overflows from a reservoir is called as water flow level of the reservoir (see Fig. 7).

Reservoir base-line: A line passing through the deepest point of a reservoir and parallel to water flow level of the reservoir is called as reservoir base-line (see Fig. 7).

Height of a reservoir: By height of a reservoir we mean the depth of water in the reservoir. In other words, height

Fig. 6. Four different mode components with their reservoirs from top, bottom, left, and right side are shown here. Reservoirs are marked by dotted region. (a) Portrait. (b) Reverse portrait. (c) Landscape. (d) Reverse landscape components.

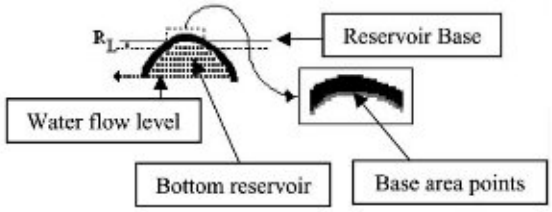

Fig. 7. Reservoir base and reservoir flow-level are shown in a component. Base-area points of the bottom reservoir are marked as grey points in a zoomed part of the component.

of a reservoir is the normal distance between reservoir base-line and water flow level of the reservoir.

Base -area points: By *base-area* points of a reservoir we mean the boarder points of the reservoir having height less than R_L from the reservoir base-line. *Base-area* points of a bottom reservoir are marked by grey pixels in Fig. 7. Here, R_L is the length of most frequently occurring horizontal black run in a component. In other words R_L is the statistical mode of the horizontal black run lengths of the component. For a component, R_L is calculated as follows. A component is scanned row-wise (horizontally). Suppose the component has n different horizontal run of lengths $r_1, r_2, \ldots\ldots r_n$ with frequencies $f_1, f_2 \ldots\ldots f_n$, respectively. Then value of R_L will be r_i if $f_i = \max(f_j)$, $j = 1, 2, \ldots\ldots n$.

IV. Mode Detection

As mentioned earlier, characters in a word are connected through head-line in Bangla and Devnagari scripts. Hence, a connected component generally represents a word in these scripts. By mode of a component, we mean the orientation of a component in a document. In our present age of technology, text lines of a document page may be written in artistic way. As a result, components in a document may have arbitrary orientations. For line extraction purposes, we classify the mode (orientation) of a component into one of the four types: portrait, reverse portrait, landscape, and reverse landscape. Examples of components of these four modes are shown in Fig. 6. Here portrait, reverse portrait, landscape, and reverse landscape components are shown in Fig. 6(a)–(d), respectively.

To detect mode of a component, we analyze reservoirs of the component obtained by pouring water from the top, bottom, left, and right side of the component. Due to the head-line in Bangla and Devnagari scripts, it is noted that many big reservoirs are created in a portrait, reverse portrait, landscape, and reverse landscape component when water is poured from the bottom, top, left, and right side of the component, respectively. For a component, we compute the areas of the reservoirs obtained by pouring water from four sides of the component. Let T^1, B^1, L^1, and R^1 be the total areas of top, bottom, left, and right reservoirs of a component, respectively. Now, we compute the biggest of these four areas. If $B^1(T^1)$ is the biggest then the mode of the component is detected as portrait [reverse portrait]. Similarly, if $L^1(R^1)$ is biggest then the mode of the component is selected as landscape [reverse landscape]. The component shown in Fig. 6(a) [Fig. 6(b)] is portrait (reverse portrait) mode because bottom (top) reservoirs have maximum area in this component. Similarly, the component shown in Fig. 6(c) [Fig. 6(d)] is landscape (reverse landscape) mode because reservoirs have maximum area when water is poured from left (right) side of the component.

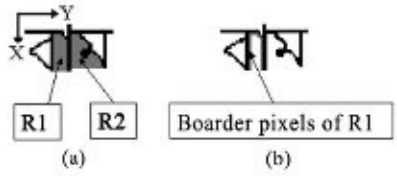

Fig. 8. (a) Two overlapping reservoirs R1 and R2 are shown. (b) Boarder pixels of the reservoir R1 are marked by grey shade.

Based on the mode of a component, the reservoirs to be used in future are chosen. For example, if the mode of a component is portrait the reservoirs obtained by pouring water from bottom will be used in future.

V. Isolated and Touching Component Detection

When two or more characters touch in Bangla or Devnagari scripts we generally observe at least one out of the three:

1) two consecutive characters create a large reservoir (for example see Fig. 2);
2) number of reservoirs in a touching component will be greater than that of an isolated component;
3) *overlapping* of two reservoirs or *overlapping* of a reservoir and a loop occurs frequently in touching characters whereas such overlapping does not exist in isolated characters. Overlapping is defined as follows.

Let B be the set containing the x coordinates of the boarder of a reservoir and B′ be the set containing the x coordinates of the boarder of another reservoir. If more than 40% of the elements of B coincide with the elements of B′ then two reservoirs are called overlapped. Examples of overlapping reservoirs are shown in Fig. 8. Computing different features like number of loops, number of reservoirs, height of the biggest reservoir, heights of the overlapping reservoirs and loops etc., obtained by the above observations, isolated, or touching component detection scheme is developed as follows.

Let for a component C

L	number of close loops;
L_i	height of ith close loop $(i = 1 \cdots L)$;
C_1	number of reservoirs obtained from C;
T	$L + C_1$;
R_i	height of the ith reservoir $i = 1 \cdots C_1$;
H	height of the component C;
R_h	$\mathrm{Max}(R_i, i = 1 \cdots C_1)$;
R'_h	$R_h + Q_i$, where Q_i $\mathrm{Max}(R_i, i$ $1 \cdots C_1$ and R_i / $R_h)$ and the reservoirs having height Q_i and R_h are nonoverlapping;
R''_h	$R_h + L_i$ where $L_i = \mathrm{Max}(L_i, i = 1 \cdots L)$ such that reservoir with height R_h and the loop with height L_i are nonoverlapping.

Now, we define the following Boolean functions based on the observations of Bangla and Devnagari touching characters.

S_- = 1 if $C_1 > 3$ or $L > 3$. Else $S_1 = 0$.
S_2 = 1 if $R_h > 50\%$ of H. Else $S_2 = 0$
S_3 = 1 if $R_h \geq 50\%$ of H or $R''_h \geq 50\%$ of H. Else $S_3 = 0$
S_4 1 if $R'_h \geq 50\%$ of H. Else S_1 0
$S_5 = 1$ if $S_3 = 1$ or $S_4 = 1$. Else $S_5 = 0$

Based on the above values the isolated and touching components detection is done by the following algorithm.

```
If T ≥ T_ then C is touching
else if T  ≥  T2 and S2 is 1 then C is
  touching
else if T  ≥  T3 and S1 is 1 then C is
  touching
else if S3 is 1 then C is touching
else if T  ≥  T4 and S4 is 1 then C is
  touching
else if T  =  T5 and S5 is 1 then C is
  touching
else C is isolated.
```

Values of different thresholds T_1, T_2, T_3, T_4 and T_5 are chosen as 8, 2, 4, 6, and 3, respectively. Based on the property of Bangla and Devnagari characters and words, values of these different thresholds are chosen [15].

VI. Line Extraction Technique

Multioriented or curved line extraction technique depends on the following three steps:

1) component type (S-type or C-type) detection;
2) candidate point computation;
3) candidate region selection.

Here, at first, we shall discuss these three steps. Next, individual text line extraction procedure will be discussed.

A. Component Type Detection

Component's type detection procedure depends on the mode of a component and the envelope points of the component. To detect the type of a component, we first find envelope points of the component from a side depending on the mode of the component. By envelope points of a component from the top we mean the set of uppermost black pixels obtained by top-down column-wise scanning of the component. If the mode of a component is portrait, reverse portrait, landscape, and reverse landscape then envelope points of the component are computed from the top, bottom, right, and left side of the component, respectively.

To compute the envelope points of a portrait (reverse portrait) component C, following task is performed. From each pixel of the top (bottom) side of the bounding box of C, we perform a vertical scan and as soon as a black (object) pixel is encountered, we label it "E." The set of "E" labeled pixels obtained in this way from top (bottom) scanning denotes the envelope points of the portrait (reverse portrait) component. Similarly, to compute the envelope points of a landscape (reverse landscape) component C, we perform horizontal scan from each pixel of the right (left) side of the bounding box of C. Envelope points on the images of four components of different modes are shown in Fig. 9. To get a clear view of the envelope points, in this figure envelope points are shown separately from their respective components. From the figure it can be noted that most of the envelope points lie on head-line portion. The points which do not lie on the head-line are part of modified characters in most cases.

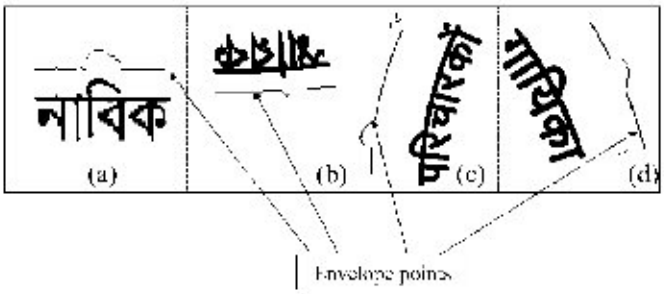

Fig. 9. Four components with their envelope points are shown. (a) Portrait. (b) Reverse portrait. (c) Landscape. (d) Reverse landscape.

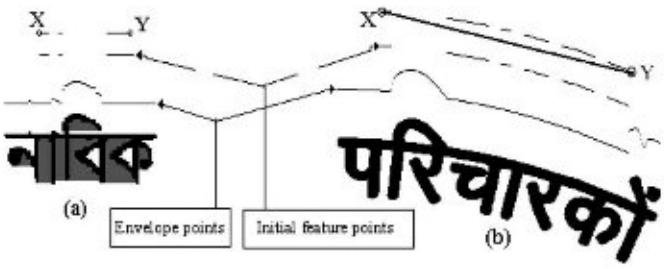

Fig. 10. Straight or curve component detection approach. (a) Example of straight component. (b) Example of curved component.

Let U_P be the set of envelope points of component C. Based on the mode of C, the reservoirs to be used for C are noted and the base-area points of these reservoirs are calculated as mentioned in Section III. As mentioned earlier, if the mode of a component is portrait only those reservoirs obtained by pouring water from bottom to be considered for that component. From these reservoirs we select reservoirs having height greater than a threshold Z. We call these selected reservoirs as *candidate reservoirs*. The value of Z is chosen as 1/10 of the corresponding component height. (This threshold value is obtained from experiment). Let B_A be the set of the base-area points of the candidate reservoirs of C. Those points of U_P, for which Euclidean distance from any point of B_A is less than R_L are noted, (R_L is computed in Section III). These points are the *initial feature points* of C. See Fig. 10 where initial feature points are shown. Based on these initial points, the straight or curve component is identified. Let X and Y be the two furthest initial points of a component. We draw a straight line passing through these two points (see Fig. 10). If the perpendicular distances of all other initial feature points of the component from the line XY are less than R_L, we identify that component as straight (S-type). Otherwise, it is identified as curve component (C-type). The component shown in Fig. 10(a) is a straight component because distance of all the initial points obtained in this component from the line XY is less than R_L. The component shown in Fig. 10(b) is a curve component because distance of all the initial points obtained in this component from the line XY is not less than R_L. The equation of straight line passing through X and Y is the estimated equation of the head-line of a straight component. For a curve component, we estimate an equation of a parabolic curve [6] passing through the initial feature points of the component. This estimated curve represents the head-line of the curve component.

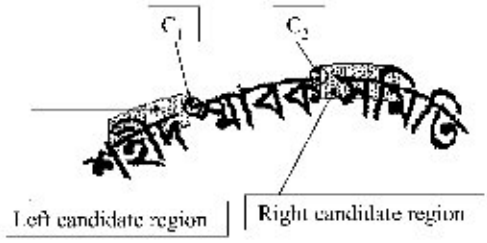

Fig. 11. Two candidate points (C_1 and C_2) of the middle component are shown by small circle. Left and right candidate regions of the middle component are shown by dotted area.

B. Candidate Point Selection

Detection of *candidate points* of a component is done based on its initial feature points and its type (C-type or S-type). For each component we detect two candidate points. For a S-type component the line XY is extended in both directions until it reaches the bounding box of the component. Two points where the extended line meets the bounding box are noted and we call these two points the candidate points. Similarly, for a curve component, estimated curve is extended in both directions to get its two candidate points. Let C_1 and C_2 be two candidate points of a component. Candidate points of a component are shown in Fig. 11.

C. Candidate Region Detection

Now, for each component we have the following:

1) component type (C-type or S-type);
2) estimated equation of the component head-line;
3) two candidate points.

For candidate region detection we use a directional region growing approach as follows. Depending on the type of a component, its estimated head-line is extended up to a distance TR ($TR = 2 \times R_h$, R_h is defined in Section V) in outwards direction from the candidate point C_1. The path obtained during this extension is called as *extended candidate path* of C_1 of the component. From each point, say P, on the extended candidate path of the component a neighborhood (a square of side R_h) is selected considering P as center. The union of the neighborhoods obtained from all points on the extended candidate path is the left candidate region of the component for the candidate point C_1. In a similar way, right candidate region of the component for the candidate point C_2 is computed. Candidate regions of a component are shown in Fig. 11.

The reason to choose TR equal to $2 \times R_h$ is as follows. In general, distance between two consecutive words in a line is less than twice the height of a character. In printed text, R_h generally represents the height of the middle zone of a word and the middle zone height of a word is equivalent to the height of a basic character. Since the distance between two consecutive words of a line is generally less than twice the height of a basic character, if we assume the value of TR as twice of R_h it is most likely that a part of a neighboring word will fall in one of the candidate regions of the current word. Based on the above observation we assume TR as $2 \times R_h$.

D. Line Extraction

For line extraction, we first cluster touching components into individual lines. Next, isolated characters are merged with the respective lines. Let α be the set of touching components in a document. We use a bottom-up approach for line extraction. The approach is as follows. First, an arbitrary component (say, top-most left component) is chosen from α and a group G is formed using this component. For each group two anchor components are maintained and we designate them as left and right anchor components. When number of component in a group is one, the left anchor component and right anchor component are the same. Since at present G has only one component, the left anchor component and right anchor component are the same. Next, we check whether there exists any component in α where at least one initial point falls into any of the candidate regions of an anchor component of the group. If there exists any component in α, we then include that component in the group G and the anchor components are modified accordingly. Let this newly included component be K. The anchor components are modified as follows. If an initial point of K falls in the left candidate region of the left anchor component, then K is assigned as left anchor component of G. Similarly, if an initial point of K falls in the right candidate region of the right anchor component then K is assigned as right anchor component of G. If two or more components fall in a candidate region of an anchor component at the same time then all these components are included in the group and the component which is farthest from the anchor component is the new anchor component. Sometimes a portion of some isolated characters may also fall in the candidate regions of an anchor component. We also include such isolated characters in the group.

Components clustered into a single group are the members of a single text line. To get other text lines we follow the same steps and finally get N number of groups if there are N text lines in the documents.

Note that most of the isolated components, which are situated between two touching components of a text line, are grouped by the above approach. Some isolated characters like dots of some characters, punctuation marks like comma, apostrophe etc. may not be included by the above approach. To include these isolated components in the group of their respective text lines we use a region growing technique [10]. An isolated component is grown along its boarder until it touches any component of a line. The isolated component is included to that line in which it touches first.

VII. Results and Discussion

For experiments, text lines were considered from different documents like magazines, newspapers, advertisements, computer printouts etc. Single and multioriented text documents, as well as curved text documents were considered for the experiment. Images were digitized by a flatbed scanner at 300 DPI. For the experiment, we considered single column document pages only. We also assume that a text line does not intersect any other text lines. From the data we noticed that number of characters in a line was between 8 to 70 characters.

TABLE I
Results on Isolated and Touching Components Detection

Component type	Identified as	
	Isolated	Touching
Isolated	98.94%	1.06%
Touching	2.06%	97.94%

TABLE II
Results on Component Type Detection

Component type	Identified as	
	Straight (S-type)	Curve (C-type)
Straight (S-type)	99.28%	0.72%
Curve (C-type)	1.19%	98.81%

Now, we shall present here results of different modules of our system.

A. Results on Isolated and Touching Component Detection

To evaluate performance of the isolated and touching component detection module a data set of 6000 components was collected from printed Bangla and Devnagari documents. Out of these 6000 components 3915 are touching and rest are isolated. The results are checked manually. The distribution of the accuracy on isolated and touching component detection is given in Table I. From the experiment we noticed that most of the errors (2.06%) came from touching components that were identified as isolated components. Length of such touching components was between two to three characters in most of the cases (length of a component is considered as the number of characters in the component). Sometimes, due to the poor quality of the documents, reservoir and loop based features are not properly detected and the desired results are not obtained in isolated and touching component detection of such documents.

B. Result on Component Type Detection

Component type detection module is tested on 3200 straight components and 2010 curved components and details experimental results on component type detection are given in Table II. From the experiment we obtained 99.28% accuracy in straight component detection. Also, we note that most of the errors occur in curve components with length two or three and these components are detected as straight components.

C. Results on Line Extraction

For experiment of line extraction module, 1521 text lines were considered from different documents. Among these 1521 text lines, 361 lines were taken from multioriented documents, 858 lines were from single oriented documents, and 302 text lines were curved. To check whether a text line is extracted correctly or not, we connect all components by line segments

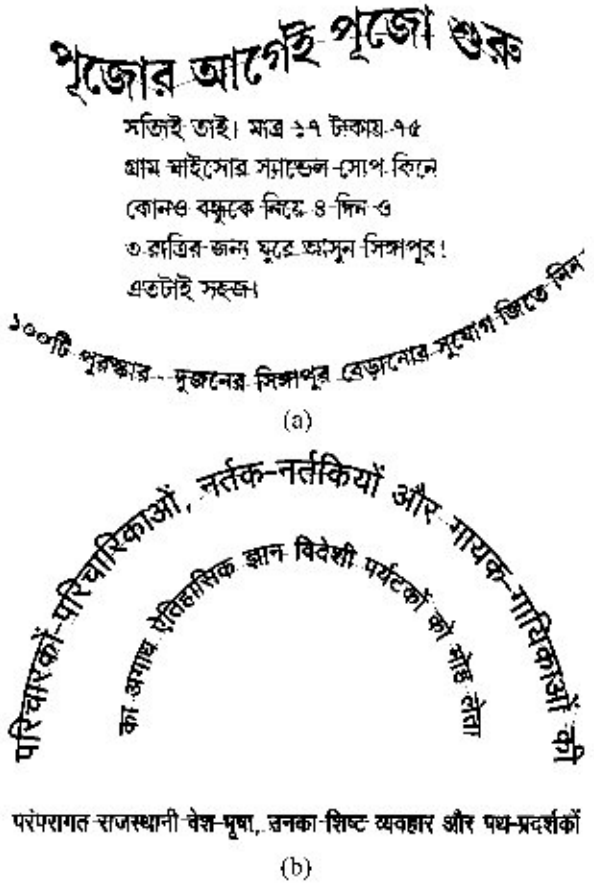

Fig. 12. Line extraction results of the proposed scheme on the documents shown in Fig. 1(a) and 1(b).

that are grouped in an individual line. These line segments are drawn through the center of gravity (CG) of the components. See Fig. 12 where results of our line extraction scheme are shown. By viewing the results on the computer's display we check the results of line extraction method manually. To give an idea about different ranges of accuracy of the system on different type documents we divide the accuracy into three stages: 100%, 96–99.9%, and <96%. Accuracy of line extraction module is measured according to the following rule. If out of N components of a line M components are extracted in favor of that line by the proposed scheme then the accuracy for that line is $(M \times 100)/N\%$. So if all components of a text line are extracted correctly by the proposed algorithm we say accuracy for the line is 100%. Distributions of results are shown in Table III. As mentioned earlier, we consider 361 text lines from multioriented documents. Out of these 361 lines we notice that on 129 text lines there is no error. We also notice that only on 29 lines the accuracy is less than 96%. For single oriented documents we note that out of 858 text lines only on 80 lines the accuracy is less than 96%.

One of the significant advantages of the proposed method is its flexibility. Our scheme is independent of font, size, and style of the text lines. For the flexibility testing, text lines with different popular fonts, sizes and styles (bold and italics) were considered. We consider font size from 8–60 points. Some standard fonts of Bangla and Devnagari scripts are used for the experiment. For example, Samit, Satyajit, and Vivek are used in Bangla, and Yogesh, Nataraj, and Surekh are used in Devnagari. The proposed method can be used in other languages which have head-line feature, e.g., the proposed method can be used in Panjabi (Gurumukhi), Marathi, and Assamese languages because they have head-line features.

TABLE III
DISTRIBUTIONS OF LINE EXTRACTION RESULTS

Document type	Accuracy obtained	On number of text lines
Multioriented	100%	129
(data size 361)	96–99.9%	203
	<96%	29
Single- oriented	100%	357
(data size 858)	96–99.9%	421
	<96%	80
Curved	100%	98
document	96–99.9%	171
(data size 302)	<96%	33

This is a new approach, to our knowledge, in this area and we hope it will give a fresh research perspective to document image analysis in Indic scripts. As there is no existing work on curve text line extraction of Bangla and Devnagari scripts, we cannot compare our results.

D. Drawbacks of the Proposed Method

Our method cannot extract a text line properly if the distance between two words of the text line is high. We assume the distance between two words of a line will be less than the height of two characters of that line. If the distance between two words is very high we have to adjust the threshold TR (discussed in Section VI) accordingly. Similarly, if two words of two different lines are very near because of artistic writing our method sometimes fails to cluster them properly. Another drawback of the proposed method is that it will not work properly if there is no character with head-line in a word. However, a word formed by characters without head-line is rare because, as mentioned earlier, from a statistical analysis we notice that the probability that a Bangla word will have head-line is 0.994 and the corresponding probability for Devnagari is 0.997 [1]. Therefore, our proposed method will work only on those scripts where head-line feature exists. As in English, there is no head-line feature the proposed method will not work on English.

REFERENCES

[1] 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.

[2] ——, "A complete printed Bangla OCR system," *Pattern Recognit.*, vol. 31, pp. 531–549, 1998.

[3] O. Deforges and D. Barba, "Segmentation of complex documents multilevel images: a robust and fast text bodied-headers detection and extraction scheme," in *Proc. 3rd Int. Conf. Document Analysis Recognition*, 1995, pp. 770–773.

[4] L. A. Fletcher and R. Kasturi, "A robust algorithm for text string separation from mined text/graphics images," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 10, pp. 910–918, Nov. 1988.

[5] H. Goto and H. Aso, "Extracting curved lines using local linearity of the text line," *Int. J. Doc. Anal. Recognit.*, vol. 2, pp. 111–118, 1999.
[6] S. C. Gupta and V. K. Kapoor, *Fundamentals of Mathematical Statistics*. New Delhi, India: Sultan Chand, 1992.
[7] A. Hashizume, P. S. Yeh, and A. Rosenfeld, "A method of detection the orientation of aligned components," *Pattern Recognit. Lett.*, vol. 4, pp. 125–132, 1986.
[8] F. Hones and J. Litcher, "Layout extraction of mixed mode documents," *Mach. Vis. Applicat.*, vol. 7, pp. 237–246, 1994.
[9] K. Kise, W. Iwata, and K. Matsumoto, "A computational geometric approach to text-line extraction from binary document images," in *Proc. IAPR Workshop Document Analysis Systems*, 1998, pp. 364–375.
[10] T. Kurita, "An efficient agglomerative clustering algorithm using a heap," *Pattern Recognit.*, vol. 24, pp. 205–209, 1991.
[11] D. S. Le, G. R. Thoma, and H. Wechsler, "Automatic page orientation and skew angle detection for binary document images," *Pattern Recognit.*, vol. 27, pp. 1325–1344, 1994.
[12] G. Nagy, S. Seth, and M. Viswanathan, "A prototype document image analysis system for technical journals," *Comput.*, vol. 25, pp. 10–22, 1992.
[13] L. O'Gorman, "The document spectrum for page layout analysis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 15, pp. 1162–1173, Nov. 1993.
[14] U. Pal, A. Belaïd, and C. Choisy, "Touching numeral segmentation using water reservoir concept," *Pattern Recognit. Lett.*, vol. 24, pp. 261–272, 2003.
[15] U. Pal and S. Datta, "Segmentation of Bangla unconstrained handwritten text," in *Proc. 7th Int. Conf. Document Analysis Recognition*, 2003, pp. 1128–1132.
[16] U. Pal, M. Mitra, and B. B. Chaudhuri, "Multi-skew detection of Indian script documents," in *Proc. 6th Int. Conf. Document Analysis Recognition*, 2001, pp. 292–296.
[17] T. Pavlidis and J. Zhou, "Page segmentation and classification," *Comput. Vis. Graph. Image Process.*, vol. 54, pp. 484–496, 1992.
[18] H. Yan, "Skew correction of document images using interline cross-correlation," *CVGIP: Graph. Models Image Process.*, vol. 55, pp. 538–543, 1993.
[19] ——, "Detection of curved text path based on the Fuzzy Curve-Tracing (FCT) algorithm," in *Proc. 6th Int. Conf. Document Analysis Recognition*, 2001, pp. 266–269.

U. Pal received the Ph.D degree in computer science from the Indian Statistical Institute, Kolkata, in 1997.

He has been with the Indian Statistical Institute since 1992 and is now a Faculty Member in the Computer Vision and Pattern Recognition Unit. In 1997, he visited GSF, Munich, Germany, for six months as a Guest Scientist where he worked on medical image analysis. In 2000, he visited INRIA, Lorraine, France for one year to pursue his post doctoral research. His fields of interest include digital document processing, optical character recognition, and medical image analysis. He has about 50 research papers on various reputed journals and conference proceedings.

Dr. Pal received the Student Best paper Award from the Computer Society of India in 1995. He also received a merit certificate from the Indian Science Congress Association in 1996. In 2003, he received ICDAR Outstanding Young Researcher Award for his significant impact in the research domain of Indian script OCR. He is a member of IUPRAI (Indian unit of IAPR), Computer Society of India, and Indian Science Congress Association.

Partha Pratim Roy was born in Kolkata, West Bengal, India in 1978. He received the B.Tech. degree in computer engineering from the Institute of Engineering and Management affiliated with the University of Kalyani, West Bengal, zIndia in 2002.

He is currently working as an Assistant System Engineer in Tata Consultancy Services, Kolkata, India.