Multimedia Tools and Applications (2019) 78:23255–23280
https://doi.org/10.1007/s11042-019-7620-6

# Drop flow method: an iterative algorithm for complete segmentation of Devanagari ancient manuscripts

Sonika Rani Narang[1] · Manish Kumar Jindal[2] · Munish Kumar[3]

Received: 10 May 2018 / Revised: 17 January 2019 / Accepted: 8 April 2019 /
Published online: 1 May 2019


**Abstract**
One of the major challenges of ancient manuscripts recognition is character segmentation. Because of many distinct features of ancient documents (thick characters, overlapping and touching characters), character segmentation is a very difficult task. Devanagari ancient manuscripts consist of vowels, consonants, modifiers, conjuncts and compound characters. Using existing techniques, segmentation of overlapping and touching characters is problematic. In this paper, an iterative character segmentation algorithm is presented for ancient documents in Devanagari script. At the beginning, the lines are extracted from the ancient documents by dividing the document image into vertical stripes and then using piecewise horizontal projection profiles. After that, these lines are segmented into words using vertical projection profiles and finally, words are segmented in characters using an iterative algorithm. In each iteration, character segmentation is refined. In the present work, we have proposed a new algorithm with the name 'Drop Flow Method' to find the segmentation path between touching components. The proposed algorithm can segment touching characters and 96.0% accuracy has been achieved for complete segmentation of Devanagari ancient manuscripts.

**Keywords** Pattern recognition · Optical character recognition · Drop flow · Ancient manuscripts · Ancient documents

## 1 Introduction

Text line, word and character segmentation are important steps in an optical character recognition system. Segmentation plays a major role in increasing the recognition accuracy of the text. It is very important to keep errors as minimum as possible in the segmentation

✉ Munish Kumar
munishcse@gmail.com

Sonika Rani Narang
sonikanarang@davcollegeabohar.com

Manish Kumar Jindal
manishphd@rediffmail.com

Extended author information available on the last page of the article

phase. Segmentation of a printed document is very easy, but not so easy for handwritten text. Especially for ancient documents, line and character segmentation are very difficult because ancient documents have many distinct features like skewed lines, touching and overlapping lines, uneven distance between consecutive lines, thick characters, uneven thickness of characters, overlapping and touching characters. Optical Character Recognition (OCR) for the Devanagari printed script has been developed up to acceptable accuracy, but OCR of handwritten Devanagari documents, especially ancient documents is still at a very early stage. Not much work has been done in this field. In this paper, the authors have proposed an iterative algorithm for segmentation of Devanagari ancient documents. First of all, lines are obtained from the ancient documents by dividing the document image into vertical stripes and then using piecewise horizontal projection profiles. Then these lines are segmented into words using vertical projection profiles and finally, words are segmented into characters using an iterative algorithm. In each iteration, character segmentation is refined and can segment any number of touching characters. Water flow method was used to segment lines from the document image [7]. Also water flow method assumes pouring of water at a certain angle. [24] used drop fall method for character segmentation. In this method, a hypothetical drop of water falls in a downward direction to find segmentation path. Tripathy and Pal [33] used the water reservoir method to segment characters. This method uses the thick flow of water, but in the present work, we have used only a single drop of water to get the water passage. In the proposed work, we have combined the above approaches to get a new technique named drop flow. The drop flow method is used to segment some touching characters. This work follows an iterative approach for the segmentation of document image. The proposed method can be used for other languages also. Based on the structure of any script, some of the iterations may be missed but overall, this approach can be used for other languages also. So, it is important for the scientific community apart from residents of India. This paper is organized as follows: Section 2 describes characteristics of the Devanagari script. Prior work is given in section 3. Section 4 describes the proposed algorithm. Results and discussions are presented in section 5. Concluding notes and future directions are given in section 6.

## 2 Characteristics of Devanagari script

Devanagari is a major script in India. Devanagari is used to write languages like Hindi, Sanskrit, Nepali and Marathi languages. Devanagari consists of 11 vowels, 33 consonants, called basic characters. Vowels may be written as independent characters or the use of diacritical marks. These diacritical marks are defined as modifiers or matras. Characters, which are derived by using modifiers, are called conjuncts. Characters, derived by combining two or more consonants, are called compound characters. A sample of Devanagari character set is given in Tables 1 and 2. In Devanagari, there is a horizontal line at the upper part of every

**Table 1** Vowels and corresponding modifiers

| Vowel | अ | आ | इ | ई | उ | ऊ | ऋ | ए | ऐ | ओ | औ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Corresponding Modifier | | ा | ि | ी | ु | ू | ृ | े | ै | ो | ौ |

**Table 2.** Consonants

| | | | | |
|---|---|---|---|---|
| क | ख | ग | घ | ङ |
| च | छ | ज | झ | ञ |
| ट | ठ | ड | ढ | ण |
| त | थ | द | ध | न |
| प | फ | ब | भ | म |
| य | र | ल | व | |
| श | ष | स | ह | |

character. This line is called as Shirorekha or headline. The shirorekha of one character adjoins with the shirorekha of the same word's next character.

Except above basic characters, Devanagari script has 3 common conjuncts. Table 3 shows common conjuncts.

There are very large numbers of conjuncts in Devanagari script. Most of the conjuncts contain 2 or 3 consonants, but there are some conjuncts which combine 4 or 5 constants. Some of the conjuncts are given in Table 4.

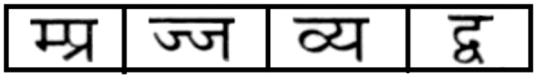
## 3 Related work

Survey papers are available for segmentation of characters from the document image [8, 11, 12, 18, 27]. Mohite and Bombade [19] discussed some challenging issues in the recognition of the Devanagari script. They used structural properties of the script and fuzzy logic based approach to the segmentation of Devanagari documents. Srivastav and Sahu [31] used horizontal and vertical projection profiles for segmentation of Devanagari documents with 90% accuracy in character segmentation. Fujisawa et al. [14] have used projection profiles for touching component segmentation with 95% accuracy. They presented a pattern-oriented segmentation method for optical character recognition that leads to document structure analysis. But, this method fails when the text is strongly skewed or overlapping. Kim et al. [17] have used contour-tracing features for segmentation. From the contour of a touching component, valley and mountain points are estimated and the cutting path is found to segment the characters. The Contour tracing algorithm does not work well if the two numerals touch in a straight-line fashion. They obtained a recognition rate of 91.8%. Chen and Wang [10] used thinning-based methods for segmenting touching handwritten numeral strings. In this paper, the combination of background and foreground analysis is used to segment touching handwritten numeral strings. Thinning of both foreground and background regions is done. Several

**Table 3** Common conjuncts

| | | |
|---|---|---|
| क्ष | त्र | ज्ञ |

**Table 4** Some of the conjuncts in Devanagari script



possible segmentation paths are constructed. Finally, the parameters of geometric properties of each possible segmentation path are determined and the best segmentation path is decided. This algorithm can get a correct rate of 96% with a rejection rate of 7.8%. But, this method is time-consuming and generates protrusions. These protrusions sometimes give wrong results. Oliveira et al. [20] used a novel segmentation approach based on contour and profile features. First, local minima of contour and profile features are defined as a basic point (BP). Second, a point with more than two pixels in its neighborhood is defined as an intersection point (IP). After that, the Euclidean distance scheme is applied to determine proximity between IP and BP. This approach cannot solve the problem of multiple touching. Bansal and Sinha [5] used the structural properties of the script like presence and relative location of a vertical line, horizontal zero crossings, number of positions of the vertex points, moments and the nature of constituent pure consonant form in the conjunct for segmentation of touching and fused Devanagari characters.

Sulem et al. [32] presented a survey of existing line segmentation methods for historical documents. They presented 6 line segmentation techniques as: projection profiles, smearing methods, grouping, Hough based methods, repulsive-attractive network and stochastic methods.

Pal et al. [21] used water reservoir based method and morphological, structural features to segment touching numerals. In this sense, if water is poured on the top or bottom, then the space will be filled with water. The distinction is performed based on the size and number of water reservoir. Bounding box (BB) is applied on touching component to find touching position. The authors considered close loops, reservoir height and distance from center of the component as features for determining segmentation points. They achieved 94.8% accuracy. Several drawbacks are reported in this work, such as, ratio of two segmented digits is too big, cutting length is very long, best reservoir did not exist, and boundary of the reservoir contains a break point. Nevertheless, the water reservoir approach might have failed when deal with broken character. Sharma and Lehal [29] proposed an algorithm to segment the words in an iterative manner by focusing on the presence of headline, aspect ratio of the characters and projection profiles. This work performed segmentation in three phases. First phase performs basic segmentation, second phase segments under-segmented words and over-segmentation is handled in the third phase. Tripathy and Pal [33] proposed water reservoir based method to segment characters of unconstrained Oriya text. At first, isolated and touching characters in a word are identified. Next touching characters of the word are segmented based on the reservoir base area points and structural feature of the component. Brodic and Milivojevic [7] presented water flow algorithm for text line segmentation. This algorithm assumes hypothetical water flows under few specified angles of the image frame from left to right and vice versa. Bar-Yosef et al. [6] proposed a novel approach for text line segmentation based on adaptive local projection profiles for degraded documents with text lines written in large skew. They applied the local algorithm in an incremental manner that adapts to the skew of each text line as it progresses. They achieved very accurate results on a set of degraded documents with lines written in different skew angles and curvature.

Jindal et al. [16] studied segmentation problems in the printed degraded Gurmukhi script and proposed solution for segmenting touching characters in the upper zone of machine printed Gurmukhi script. This technique is based on the structural properties of the Gurmukhi script characters. Concavity and convexity of the characters have been studied and using top profile projections, the touching characters in upper zone have been segmented. The recognition rate of 91% has been achieved for segmenting the touching characters in the upper zone. Alam and Kashem [2] gave a complete Optical Character Recognition (OCR) system for printed Bangla characters. They used the headline and baseline to segment characters. Reddy et al. [26] used topological properties in terms of zones, component combinations, and behavioral aspects of syllables in the segmentation process for Telugu script. They proposed split profile algorithm while handling touching components. Alaei et al. [1] used a piecewise painting technique for line segmentation of unconstrained handwritten text. They decomposed text block vertically into parallel pipe structures called as strips. Each row in each strip is painted with a gray intensity, which is the average intensity value of the gray values of all pixels present in that row-strip. Subsequently, painted pipes are converted into two-tone paint and smoothed. They found piece-wise Potential Separating Line (PPSL) between two consecutive black space. The PPSLs are concatenated to produce the segmentation of text lines. Sridevi and Sbashini [30] have used computational intelligence techniques for text line and character segmentation of Tamil ancient documents. In this work, two methods are proposed, one for line segmentation and another for character segmentation. First method uses projection profile and PSO for line segmentation. In the second method combination of connected components along with the nearest neighborhood methods are used to segment the characters.

Shah et al. [28] used neighborhood tracing algorithm and projection profile in order to provide individual characters from a word in the handwritten Devanagari text. Panichkriangkrai et al. [23] have proposed text and line extraction system for Japanese historical woodblock printed books. Vertical projection was used on binarized images for separating text lines. Connected components (CC) were extracted by applying an adaptive binarization on the grayscale images. Rule-based integration was applied to merge or split the connected components to extract characters. Gatos et al. [15] presented a novel segmentation module for text zone as well as a text line detection for handwritten ancient documents to handle several challenging cases such as horizontal and vertical rule lines overlapping with the text, two column documents and characters of different text lines touching vertically. Bag and Krishna [4] proposed segmentation of characters in handwritten Hindi words based on structural patterns. The proposed method can cope with high variations in writing style and skewed header lines as input. The average success rate is 96.93%. Rao et al. [25] have used a hybrid model that entails segmentation in noisy images followed by binarization for segmentation of ancient Telugu documents. In the first phase, the horizontal profile pattern is convolved with a Gaussian kernel. The statistical properties of meaningful units are explored through an extensive analysis of the geometrical patterns of meaningful units. In the second phase, noisy documents are cleaned with the help of a modified IGT (Iterative Global Threshold) algorithm and then segmented by using the conventional profile mechanism. They obtained maximum accuracy of 95.59% for the cleaned story books. Chen et al. [9] have presented a Conditional Random Field (CRF) model to segment handwritten historical document images into different regions. Page segmentation was considered as a pixel-labeling problem. Features were learned from pixel intensity values with stacked convolutional auto-encoders in an unsupervised manner. Then a CRF model was introduced to improve the segmentation. As per the best of our knowledge, not much work has been done for

segmentation of Devanagari ancient manuscripts. In this paper, we propose a scheme to segment characters using an iterative approach in preprocessed binarized Devanagari ancient manuscripts. Babu and Jangid [3] presented an algorithm for segmentation of touching characters in Devanagari script using structural properties of the script. They achieved the accuracy of 85%. Dutta et al. [13] released a new handwritten word dataset for Devanagari, IIIT-HW-Dev to alleviate some of the challenging issued in Devanagari script. They benchmarked the IIIT-HW-Dev dataset using a CNN RNN hybrid architecture. They used the proposed pipeline on a public dataset, RoyDB and achieve state of the art results. Water flow method was used to segment lines from the document image [7]. Also water flow method assumes pouring of water at a certain angle. [24] used drop fall method. In this method, a hypothetical drop of water falls in a downward direction to find segmentation path. Tripathy and Pal [33] used the water reservoir method to segment characters. This method uses the thick flow of water, but in the present work, we have used only a single drop of water to get the water passage. In the proposed work, we have combined the above approaches to get a new technique named drop flow. The drop flow method is used to segment some touching characters.

## 4 Proposed work

A typical OCR system consists of a number of phases like image acquisition, pre-processing, segmentation, feature extraction, and classification. In the present work, we have proposed algorithm for the third phase of OCR (segmentation).

### 4.1 Digitization and pre-processing

For experimental work, Devanagari ancient documents were collected from various libraries and museums. These documents were preprocessed to get clean black and white images. For image preprocessing, 3 steps are taken. First, document image is enhanced by using an autocorrect feature of office image viewer. As a second step image is converted into a binary image and global threshold value is considered for binarization. If the global threshold value does not provide good results, then the local threshold value is considered as the third step.

### 4.2 Segmentation

Segmentation phase is used to segment the input document image into lines, words and characters. The algorithms used for segmentation are given below:

#### 4.2.1 Line segmentation

Line segmentation is the process of identifying lines from a document image. Devanagari ancient documents have slanting and touching/overlapping lines. We used piecewise projection profile to segment lines [1]. The document image is divided into vertical strips and then piecewise horizontal projection profile was used to segment lines. Next average line height was used to check the correctness of segmentation. Based on average line height some lines were found to be over-segmented and some lines were found to be under-segmented. Under-segmentation and over-segmentation were dealt with in the succeeding phase.

A general algorithm for line segmentation is given below:
*Algorithm* line_segmentation.

| | |
|---|---|
| Step I. | Divide the document image in stripes of fixed size. |
| Step II. | Compute horizontal projection profiles (HPP) of each row of each stripe. |
| Step III. | If HPP<=2 for any row, then that row is considered as a piece-wise separating line (PSL). |
| Step IV. | Consecutive PSLs are reduced to one PSL only. |
| Step V. | Average line height(avg_line_height) is computed. |
| Step VI. | Based on avg_line_height, over-segmentation is detected and handled. |
| Step VII. | Based on avg_line_height, under-segmentation is detected and handled. |
| Step VIII. | Finally, lines are separated. |

The results of this algorithm are shown in Figs from 1 to 5. Figure 1 depicts binarized and preprocessed Devanagari ancient document. Line segmentation using piecewise projection profile is depicted in Fig. 2. Figure 3a shows over segmented components and Fig. 3b shows results after handling over-segmentation. An under segmented component and the search area for missing PSL is depicted in Fig. 4. Figure 5a shows under segmented components and Fig. 5b shows results after handling under-segmentation.

Complete results of line segmentation algorithm are shown in Fig. 6. In Fig. 6 different lines of the document are shown in different colours.

**Limitations of line segmentation algorithm** Line segmentation algorithm depends on the average line height. If this is computed wrong, the results may be wrong as shown in Fig. 7b. Figure 7a gives the original document and Fig. 7b shows the document after line segmentation. As we can see that lines are not correctly segmented. This is because of wrong average line height.

### 4.2.2 Word segmentation

To segment words from segmented lines, the vertical projection profile method is used [33]. If we get consecutive columns with 0 black pixels, then it is decided to be a word boundary.

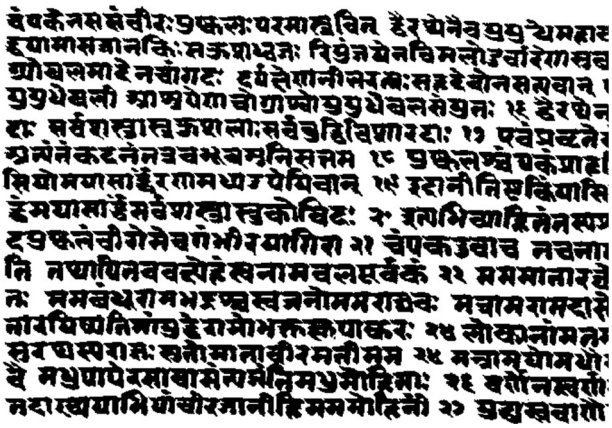


**Fig. 1** Binarized Devanagari Ancient Manuscript

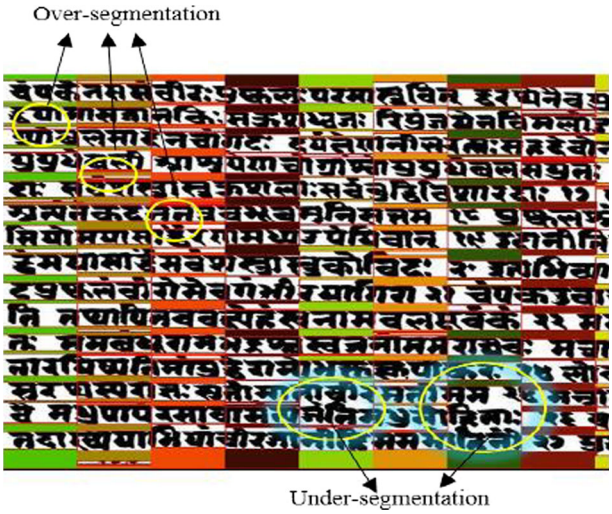
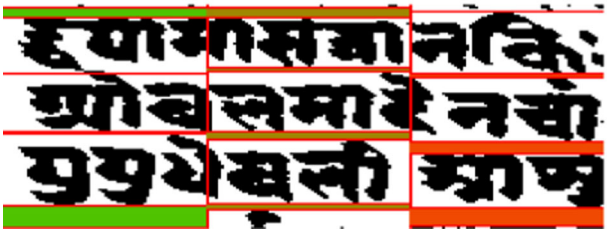



**Fig. 2** Line segmentation using Piece wise projection profile

### 4.2.3 Character segmentation

Character segmentation of words is difficult because (i) two consecutive characters of a word may touch each other (ii) two side-by-side non-touching characters may overlap. They may not be vertically separable. Character segmentation in Devanagari documents becomes very easy if the headline (Shirorekha) is removed, but ancient Devanagari documents have thick and uneven headline. So, it is very difficult to remove the headline from such documents. Characters are segmented without removing headline. The process of character segmentation using multiple iterations are illustrated in Fig. 8.

**(a)** Over-segmented components shown in circles

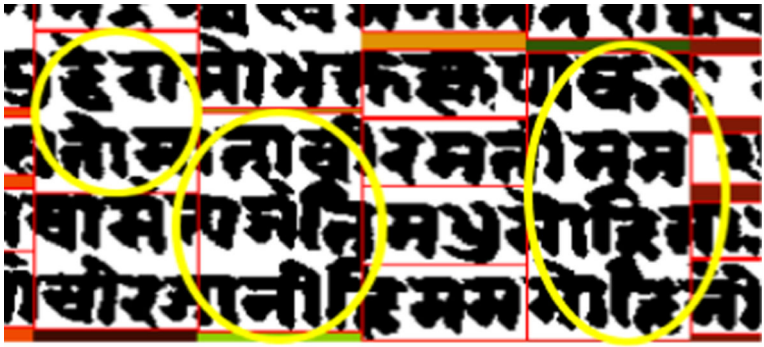


**(b)** Document after resolving over-segmentation

**Fig. 3** **a** Over-segmented components shown in circles. **b** Document after resolving over-segmentation.

**Fig. 4** Search area for missing PSL in under-segmented lines

As depicted in Fig. 8, characters of the document image are segmented in multiple iterations. In first iteration, connected components in the document image are found. Connected components of the sample document are shown in Fig. 9. All the connected components were assigned a different colour. Due to writing style of ancient documents, many times, most of the characters have been



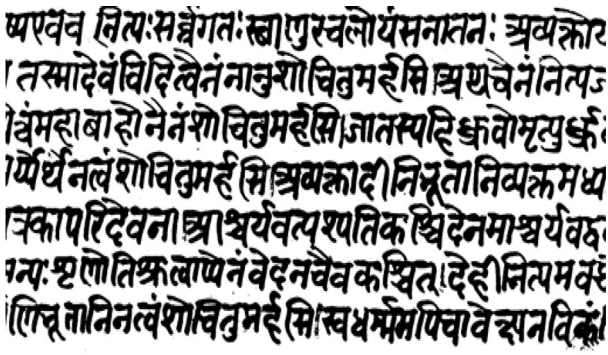
**(a)** Under-segmented components shown in circles

**(b)** Document after resolving under-segmentation

**Fig. 5** **a** Under-segmented components shown in circles. **b** Document after resolving under-segmentation.

Fig. 6 Different lines are shown in different colours

segmented correctly as in most of the ancient documents, there is a slight break in the headline after every character as depicted in Fig. 10. But, there are many characters which are connected with the headline as shown in Fig. 11.



**(a)** Sample document

**(b)** Incorrect line segmentation

Fig. 7 **a** Sample document. **b** Incorrect line segmentation

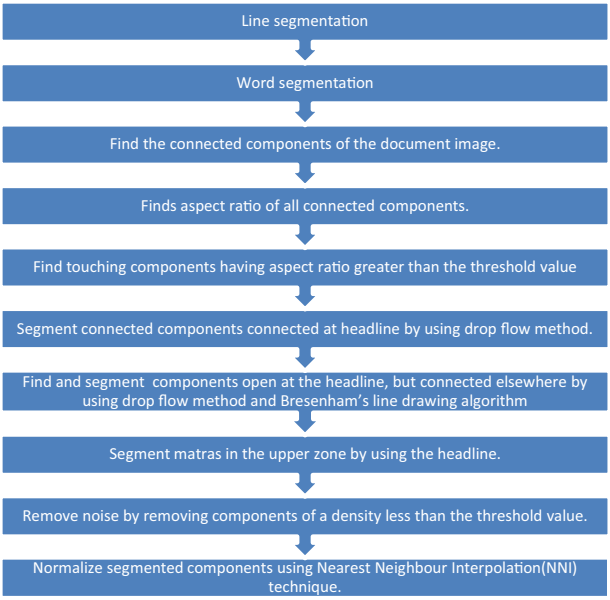



**Fig. 8** Character segmentation process

Also, there are other types of components which are not connected with the headline, but are connected somewhere else. Figure 12 shows one such component.

To find touching/overlapping characters, aspect ratio of all the connected components was found. To find the aspect ratio we recorded the coordinates of the start and end of each connected component.

$$\text{aspect}[l][i] = \frac{\text{end\_char}[l][i] - \text{start\_char}[l][i])}{(\text{line\_end\_boundary}[s][l] - \text{line\_start\_boundary}[s][l])}$$

Here, aspect[l][i] is the aspect ratio of $i^{th}$ component in $l^{th}$ line.

| | |
|---|---|
| end_char[l][i] | is the maximum $x$ coordinate of the $i^{th}$ component in $l^{th}$ line |
| start_char[l][i] | is the minimum $x$ coordinate of the $i^{th}$ component in $l^{th}$ line |
| line_end_boundary[s][l] | is the maximum $y$ coordinate of $l^{th}$ line of $s^{th}$ strip |
| line_start_boundary[s][l] | is the minimum $y$ coordinate of $l^{th}$ line of $s^{th}$ strip |

If the aspect ratio of any component is greater than the threshold value, it is identified as having touching characters. This threshold value is determined after thorough experimentation with different values. Figure 13 shows connected components of our sample document image. Different connected components are shown in different colours. Components with touching characters are shown in bounding boxes in Fig. 13. An example of a component with touching characters is shown in Fig. 14a. As we can see that these characters are connected by the headline only. In the second iteration, such touching characters were segmented.

In ancient documents, the headline is thick and uneven. So, headline removal algorithms don't work well with such documents. We tried to segment characters without removing headline. Water flow method was used to segment lines from the document image [7]. Also water flow method assumes pouring of water at a certain angle. [24]

**Fig. 9** Connected component of our sample document

used drop fall method. In this method, a hypothetical drop of water falls in a downward direction to find segmentation path. Tripathy and Pal [33] used the water reservoir method to segment characters. This method uses the thick flow of water, but in the present work, we have used only a single drop of water to get the water passage. In the proposed work, we have combined the above approaches to get a new technique named drop flow. The drop flow method is used to segment some touching characters. For this, first of all the presence of headline was estimated based on the maximum number of pixels in the horizontal direction in the upper half of the connected component. To drop flow method, a hypothetical drop of water was poured from the bottom towards the upper side. Water would find its path if there was a white pixel just above or to the above left or to the above right of the current pixel. If the water is able to find its path up to the headline, then that path would be considered as the segment path and the headline was segmented at that point. Figure 14b shows the segmented components after using drop flow method of Fig. 14a. There are few problems with this method. Sometimes, though very rarely, maximum number of pixels in a row is not in the headline.

In Fig. 15a, maximum number of pixels is in the vertical middle of the component. Such situation gives bad results as can be seen in Fig. 15b. There are some characters which are

**Fig. 10** Characters already segmented because of writing style

**Fig. 11** Characters connected at headline

**Fig. 12** Characters not connected at headline but somewhere else

connected by 1, 2 or 3 consecutive pixels at a place. Based on the thickness of the character, it was observed that such connection between characters was due to noise in the document. Such touching components were also segmented using drop flow method, but now 1, 2 or 3 black pixels (based on thickness of characters) on the way were ignored and were changed to background colour. One such component is illustrated in Fig. 16a and result after segmentation of this component is shown in Fig. 16b.

In the next iteration, those components were considered which were open in the headline but connected somewhere else. To segment this type of connected components, flow of a drop is found from top to bottom in addition to the flow from bottom to top. The point where the water stops was found in both the flows. Then Bresenham's line drawing algorithm was used to find the segmentation path between these two points. One such component is shown in Fig. 17a and result after segmentation is shown in Fig. 17b.

Up to this iteration, we got vertical segmentation paths. But, in Devanagari characters are divided in the top strip, core strip and the bottom strip. It was observed that in most of the ancient documents, there is no clear separation between core strip and the bottom strip. But, we can separate the top strip and core strip. For this work, we have considered only two strips: top strip and core strip as shown in Fig. 18.

Up to this point, matra (modifier) in the top strip is not segmented. Matras in top strip are segmented in the next iteration. To achieve the previous results, presence of headline was estimated based on the maximum number of black pixels in the upper half of the character. In these documents, the headline is a thick line. So, the line just above the headline may not be the required segmentation line. From the headline, we continue

**Fig. 13** Touching components in bounded boxes

**(a)** Characters connected at headline only

**(b)** Result after using water flow method

**Fig. 14** **a** Characters connected at headline only. **b** Result after using water flow method

moving in an upward direction until we get a line where difference of horizontal projection profile between the headline and the current line is greater than the threshold value. This line is considered to be the required segmentation line between matra and character. Results of the matra segmentation are shown in Fig. 19.

There are still some connected components which need to be segmented such as shown in Fig. 20.

We will treat some of these components as conjuncts. In the process of segmentation, we got some small dots as noise. These dots were removed in the next iteration. To remove these dots, number of pixels in each character was counted. Characters with pixels less than the threshold value were removed from the segmented image as shown in Fig. 21. The threshold value is decided after experimentation.

Final document after character segmentation is shown in Fig. 22. Different characters are shown in different colours. In the next iteration separate image of each character was created. These images were normalized into a 64 × 64 window using Normalization Nearest Neighbour Interpolation (NNI) algorithm. Figure 23 shows finally separated characters.

Based on above discussion, character segmenting algorithm may be concluded as follows:

**(a)** **(b)**

**Fig. 15** Result using water flow method **a** A sample where maximum black pixels are not in the headline **b** Characters not segmented correctly

**Fig. 16** **a** Component before segmentation **b** after segmentation

**Algorithm for character_segmentation** This algorithm gives detailed steps for character segmentation. Following data structure has been used in the algorithm:

aspect[l][i] is the aspect ratio of $i^{th}$ component in $l^{th}$ line

end_char[l][i] is the maximum x coordinate of the $i^{th}$ component in $l^{th}$ line

start_char[l][i] is the minimum x coordinate of the ith component in $l^{th}$ line

line_end_boundary[s][l] is the maximum y coordinate of lth line of $s^{th}$ strip.

line_start_boundary[s][l] is the minimum y coordinate of lth line of sth strip

con_startx[c] = minimum x coordinate of the cth touching component,

con_starty[c] = minimum y coordinate of the cth touching component,

con_endx[c] = maximum x coordinate of the cth touching component,

con_endy[c] = maximum y coordinated of the cth touching component,

hpp = horizontal projection profile

maxhpp = maximum value of hpp

heady = y coordinate of the component having maximum number of pixels in that row.

hpp_prev = hpp of previous lines

cc[y][x] = a number representing the intensity code of pixel(x, y). For a white pixel, cc[y][x]=0 and for any other colour cc[y][x] != 0

Step 1: Find connected components of the document image.
Step 2: Find aspect ratio of all connected components.

To find the aspect ratio we recorded the coordinates of the start and end of each connected component.

$$\text{aspect}[l][i] = \frac{\text{end\_char}[l][i] - \text{start\_char}[l][i])}{(\text{line\_end\_boundary}[s][l] - \text{line\_start\_boundary}[s][l])}$$

**Fig. 17** **a** Component before segmentation **b** after segmentation

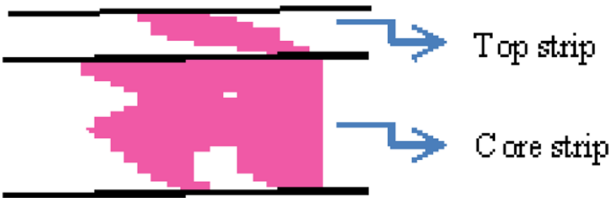



**Fig. 18** Two strips of Ancient Devanagari word

Step 3: Find touching character components.

If aspect ratio of any connected component is greater than a certain threshold value, then this connected component is assumed to be having touching characters. Then, coordinates of this touching component are recorded as con_startx, con_starty, con_endx and con_endy.

```
if (aspect[l][i]>1.3) then
        con_startx[c]=start_char[l][i]
        con_starty[c]=y_min(start_char[l][i],end_char[l][i],l)
        con_endx[c]=end_char[l][i]
        con_endy[c]=y_max(start_char[l][i],end_char[l][i],l)
end if
```

Here y_min is a function used to find the minimum y coordinate of the component.and

y_max is a function used to find maximum y coordinate of the component.

These functions are required because one touching component may span over more than one stripes and each stripe may have different minimum y and maximum y values.

Step 4: Estimate the position of headline of each touching component (a line with maximum number of black pixels).

**(a)** **(b)**

**Fig. 19** **a** Component before matra segmentation **b** after segmentation

```
for(y=con_starty[i] to con_starty[i]+(con_endy[i]-con_starty[i])/2)  //upper half of the line
        {
                hpp=0;
                for(x=con_startx[i] to con_endx[i])
                        if(cc[y][x]!=0) then
                                hpp = hpp+1
                        end if
                end for
                if(hpp>maxhpp) then
                        maxhpp=hpp;
                        heady=y;
                end if
        end for
```

Step 5: Find those connected components which are connected with headline only or connected anywhere by at most 3 pixels by using drop flow method from bottom to upward direction and segment those.

**Fig. 20** Components which are not correctly segmented

**(a)** Component with noise

**(b)** Component after noise removal

**Fig. 21** **a** Component with noise. **b** Component after noise removal

```
for(x=con_startx[i]+1 to x<con_endx[i]-1)
 x1=x
 for(p=1 to 10)
        onwaypixelx[p]=0
        onwaypixely[p]=0
 p=0
 line_status=0
 for(y = con_endy[i] to heady+7 step -1)
        if(cc[y][x]!=0) then
          if(cc[y-1][x]=0) then
                onwaypixelx[p]=x
                onwaypixely[p]=y
                p = p+1
                y = y-1
           else
                if(cc[y][x-1]!=0) then
                   if(cc[y][x+1]!=0) then
                          line_status=1
                          break
                   else
                          x+1
                   end if
                else
                   x-1
            end if
          end if
         end if
        end for
   if(line_status=0) then
         for(y=heady-5 to y=heady+5)
                cc[y][x]=0
                if(onwaypixelx[0]!=0) then
                 for(temp=0 to temp=p)
                        cc[onwaypixelx[temp]][onwaypixely[temp]]=0
                 end for
                end if
          end for
        end if
  x=x1
end for
```

Step 6: Find those components which are open in the headline but connected elsewhere by using drop flow method in an upward as well as downward direction and then use Bresenham's line drawing algorithm to segment these components.

```
[steps for drop flow in downward direction]
for x=con_startx[i]+1 to con_endx[i]
        x1=x;
        line_status=0;
        for y = con_starty[i] to con_endy[i]
          if(cc[y][x]!=0)
                if(cc[y][x+1]!=0)
                  if(cc[y][x-1]!=0)
                        if(y>heady+4)
                           line_status=1
                        else
                           line_status=2
                           break;
                        end if
                  else
                        x = x-1
                  endif
                end if
          else
                x = x+1
          end if
        end for
end for
```

Step 7: Segment matras in the upper zone by using the headline.

From the headline, we continue moving in an upward direction until we get a line where difference of horizontal projection profile between the headline and the current line is greater than the threshold value. This line is considered to be the required segmentation line between matra and character.

```
For y=heady-1 to ystart_char[l][i]
        if(maxhpp-hpp_prev[y-ystart_char[l][i]]>5) then
                reqd_y=y
        endif
end for
```

Step 8: Remove noise by removing components with density less than threshold value.

Step 9: Normalize segmented components using Nearest Neighbourhood Interpolation (NNI) technique.

## 5 Results and discussion

### 5.1 Results for line segmentation

For experiments of our algorithm, Devanagari ancient manuscripts were binarized and preprocessed. Those images were selected which did not have partial lines. The proposed line segmentation algorithm was tested on 1500 text lines from 130 document images. These lines were taken from different manuscripts with different writing styles. To check whether a line is segmented correctly or not, we generated a coloured image of the document with a different colour for each line (as shown in Fig. 6). Accuracy of line segmentation algorithm is measured according to the following rule: If $X$ out of $Y$ characters falls correctly in their respective lines, then the accuracy of the algorithm is: $X/Y \times$ 100%. As observed, accuracy obtained for line segmentation in these documents is between 97.0% and 100%. For many documents, line segmentation accuracy is 100%. Most of the errors in segmentation are observed because of the modifier ( ) in the upper zone. Many times, this modifier is not connected with the

**Fig. 22** Final result of our sample document after character segmentation

**Fig. 23** Separated and normalized characters of our sample document

headline. So it may be seen as part of the previous line. At least 97.0% of the characters fall correctly in their respective lines. This accuracy can be enhanced in the next phases of OCR.

### 5.2 Results for character segmentation

Ancient manuscript contains isolated as well as touching/overlapping characters. To check whether a character is segmented correctly or not, we generated a coloured image of the document with a different colour for each character (as shown in Fig. 22). Accuracy of the character segmentation algorithm is measured according to the following rule: If $X$ out of $Y$ characters is segmented correctly, then the accuracy of the algorithm is: $X/Y \times 100\%$.

### 5.3 Results for isolated components

For isolated characters average accuracy of the proposed algorithm is 98.5%. An error occurs when the width of a touching component is small and it is identified as an isolated character or a character is over segmented.

### 5.4 Results for overlapping/touching components

Our algorithm can handle overlapping components very well. The average accuracy of the proposed algorithm for overlapping components is considered to be 100%. In Devanagari ancient manuscripts, there are two characters touching or more character touching components. Our algorithm can deal with any number of touching characters. Accuracy of our proposed algorithm for touching characters is about 96.0%. Our algorithm can segment components which touch at multiple places. Our proposed

**Fig. 24** Component with small width of touching characters

**Fig. 25** Over-segmented characters due to large width of isolated character

algorithm is independent of character size, but depends somewhat on the thickness of the character. There is no requirement of normalization for character segmentation. We have proposed normalization for feature extraction phase. An error occurs if the touching area of a connected component is large. Also, an error occurs when the width of a touching component is small and it is identified as an isolated character (Fig. 24) or the width of isolated character is large and the character is over segmented (Fig. 25).

### 5.5 Comparison with existing work

As no standard database is available for ancient Devanagari script and no literature is available for complete segmentation of the same. So, we have tried to refer that work in result analysis which is as close to our work as possible. In future, we are planning to benchmark our own database. Accuracy comparison of proposed work with existing techniques is provided in Table 5.

## 6 Inferences and future directions

An algorithm for character segmentation in Devanagari ancient manuscripts is proposed in this paper. Here, at first, the piecewise horizontal projection profile was used for identifying lines. Next based on average line height, under-segmentation and over-segmentation was handled. After that, the words were segmented using vertical projection profiles. For character segmentation, a novel technique 'drop flow method' is proposed. Also, connected component analysis, Bresenham's line algorithm and other

**Table 5** Comparison with existing methodologies

| Author | Data | Technique | Accuracy |
|---|---|---|---|
| Tripathy and Pal [33] | Oriya Handwritten Text | Water Reservoir method | 95.0% |
| Palakollu et al. [22] | Devanagari Handwritten Text | Straighten Headline | 89.9% |
| Rao et al. [25] | Telugu Ancient Documents | Hybrid model | 70.6% |
| Proposed Work | Ancient Devanagari Text | Drop Flow Algorithm | 96.0% |

structural features are used in the segmentation process. Then a noise was removed and nearest neighbour interpolation method was used for normalization of characters. We achieved an overall accuracy of 96% for character segmentation which is better than other proposed techniques in the literature. To the best of our knowledge, this is the first work of its kind on Devanagari ancient manuscripts. The proposed algorithm can be used for some other Indian scripts also which are similar in structure to Devanagari. The proposed work has following limitations: This algorithm depends a lot on average line height. If it is not computed correctly, then our algorithm may not work correctly as depicted in Fig. 7a and b. Also, this algorithm may not work well with manuscripts having partial lines. Some conjuncts and very complex connected components are not segmented correctly by this algorithm as shown in Fig. 20. This algorithm does not work if the maximum number of pixels is not in the headline as depicted in Fig. 15a and b. This method may result in less accurate if the width of the touching area is more or the width of the connected component is less. Some of such components will be considered as conjuncts while some components occur due to errors (Fig. 20). Also, an error occurs when the width of a touching component is small and it is identified as an isolated character (Fig. 24) or the width of isolated character is large and the character is over segmented (Fig. 25). Future work may involve dealing with these limitations.

## References


1. Alaei A, Nagabhushan P, Pal U (2011) Piece-wise painting technique for line segmentation of unconstrained handwritten text: a specific study with Persian text documents. Pattern Anal Applic 14(4):381–394
2. Alam MM, Kashem MA (2010) A complete Bangla OCR system for printed characters. International Journal of Computer and Information Technology 1(1):30–35
3. Babu S, Jangid M (2016) Touching character segmentation of Devanagari script. ICCCNT '16 Proceedings of the 7th International Conference on Computing Communication and Networking Technologies, Article No. 26, Dallas, TX, USA: doi:https://doi.org/10.1145/2967878.2967908
4. Bag S, Krishna A (2015) Character segmentation of Hindi unconstrained handwritten words. Proceedings of the 17th International workshop on Combinatorial Image Analysis 9448:247–260
5. Bansal V, Sinha RMK (2002) Segmentation of touching and fused Devanagari characters. Pattern Recogn 35(4):875–893
6. Bar-Yosef I, Hagbi N, Kedem K, Dinstein I (2009) Line segmentation for degraded handwritten historical documents. 10th International Conference on Document Analysis and Recognition, Barcelona, pp 1161–1165
7. Brodic D, Milivojevic Z (2009) Reference text line identification based on water flow algorithm. Proceedings of the International Scientific Conference on Information, Communication and Energy Systems and Technologies 17(1):30–47
8. Casey RG, Lecolinet E (1996) A survey of methods and strategies in character segmentation. IEEE Trans Pattern Anal Mach Intell 18(7):690–706
9. Chen K, Seuret M, Liwicki M, Hennebert J, Liu CL, Ingold R (2016) Page segmentation for historical handwritten document images using conditional random fields. Proceedings of the International Conference on Frontiers in Handwriting Recognition (ICFHR), 90–95
10. Chen Y, Wang J (2000) Segmentation of single-or multiple-touching handwritten numeral string using background and foreground analysis. IEEE Trans Pattern Anal Mach Intell 22(11):1304–1317
11. Dogra S, Sehgal A (2017) Devanagari letters segmentation and recognition system: a brief review. IJSRD - International Journal for Scientific Research & Development 5(01):1418–1422
12. Dunn CE, Wang PSP (1992) Character segmentation techniques for handwritten text-a survey. Proceedings of the 11th International Conference on Recognition Methodology and Systems 2:577–580
13. Dutta K, Krishnan P, Mathew M, Jawahar CV (2018) Offline Handwriting Recognition on Devanagari Using a New Benchmark Dataset. 13th IAPR International Workshop on Document Analysis Systems (DAS), Vienna, pp 25–30. https://doi.org/10.1109/DAS.2018.69

14. Fujisawa H, Nakano Y, Kurino K (1992) Segmentation methods for character recognition from segmentation to document structure analysis. Proc IEEE 80(7):1079–1092
15. Gatos B, Louloudis G, Stamatopoulos N (2014) Segmentation of historical handwritten documents into text zones and text lines. Proceedings of the International Conference on Frontiers in Handwriting Recognition (ICFHR), 464–469.
16. Jindal MK, Lehal GS, Sharma RK (2009) Segmentation of touching characters in upper zone in printed Gurmukhi script. Proceedings of 2nd Bangalore Annual Compute Conference, 1–6
17. Kim KK, Kim JH and Suen CY (2000) Recognition of unconstrained handwritten numeral strings by composite segmentation method. Proceedings of the 15th International Conference on Pattern Recognition: 594–597
18. Kumar A, Yadav M, Patnaik T, Kumar B (2013) A survey on touching character segmentation. International Journal of Engineering and Advanced Technology 2(3):569–574
19. Mohite RS, Bombade BR (2014) Challenging issues in Devanagari script recognition. International Journal Computer Technology & Applications 5(3):947–952
20. Oliveira LS, Lethelier E, Bortolozzi F, Sabourin R (2000) A new approach to segment handwritten digits. Proceedings of the 7th International Workshop on Frontiers in Handwriting Recognition, 577–582
21. Pal U, Belaid A, Choisy C (2003) Touching numeral segmentation using water reservoir concept. Pattern Recogn Lett 24(1–3):261–272
22. Palakollu S, Dhir R and Rani R (2012) Handwritten Hindi text segmentation techniques for lines and characters. Proceedings of the World Congress on Engineering and Computer Science, 1–5
23. Panichkriangkrai C, Li L and Hachimura K (2013) Character segmentation and retrieval for learning support system of Japanese historical books. Proceedings of the 2nd International Workshop on Historical Document Imaging and Processing, 118–122
24. Rao S, Junitha M, Bhaskara S, Rao S (2014) Segmentation of touching Telugu characters under Noisy environment. Journal of Emerging Trends in Computing and Information Sciences 5(9):698–702
25. Rao NV, Sastry ASCS, Chakravarthy ASN, Rao AVS (2015) Analysis of canonical character segmentation technique for ancient Telugu text documents. J Theor Appl Inf Technol 82(2):311–320
26. Reddy LP, Babu TR, Rao NV, Babu BR (2010) Touching syllable segmentation using Split profile algorithm. International Journal of Computer Science Issues 7(3):17–26
27. Saba T, Sulong G, Rehman A (2010) A survey on methods and strategies on touched character segmentation. International Journal of Research and Reviews in Computer Science 1(2):103–114
28. Shah K, Singh J, Pushkarna P, Kurawadwala H, Alate A (2013) A new approach for segmentation of Devnagari characters. Global Journal for Research Analysis 2(4):162–164
29. Sharma DV, Lehal GS (2006) An iterative algorithm for segmentation of isolated handwritten words in Gurmukhi script. The 18th International Conference on Pattern Recognition (ICPR'06), pp 1022–1025
30. Sridevi N, Sbashini P (2012) Segmentation of text lines and characters in ancient Tamil script documents using computational intelligence techniques. Int J Comput Appl 52(14):7–12
31. Srivastav A, Sahu N (2016) Segmentation of Devanagari handwritten characters. Int J Comput Appl 142(14):15–18
32. Sulem LL, Zahour A, Taconet B (2007) Text line segmentation of historical documents: a survey. Int J Doc Anal Recognit 9(2–4):123–138
33. Tripathy N, Pal U (2006) Handwriting segmentation of unconstrained Oriya text. SADHANA 31(6):755–769

**Sonika Rani Narang** received her Bachelor's degree in Inter Arts in 1993 and Post Graduate degree in Computer Applications from Guru Nanak Dev University, Amritsar, India in 1996. She is pursuing her PhD degree in computer science from Panjab University, Chandigarh, India. She is working as Assistant Professor in DAV College, Abohar, Punjab, INDIA. Her research interests include Character Recognition.

**Manish Kumar Jindal** received his Bachelors degree in science in 1996 and Post Graduate degree in Computer Applications from Punjabi University, Patiala, India in 1999. He holds a Gold Medal in his Post graduation. He received his Ph.D. degree in Computer Science & Engineering from Thapar University, Patiala, India in 2008. He is working as Associate Professor in Panjab University Regional Centre, Muktsar, Punjab, India. His research interests include Character Recognition.

**Munish Kumar** received his Master's degree in Computer Science & Engineering from Thapar University, Patiala, India in 2008. He received his Ph.D. degree from Thapar University, Patiala, India in 2015. He started his career as an Assistant Professor in computer application at Jaito Centre of Punjabi university, Patiala. Presently, he is working as Assistant Professor in Department of Computational Sciences, Maharaja Ranjit Singh Punjab Technical University, Bathinda, Punjab, India. His research interests include Character Recognition, Computer Vision and Pattern Recognition.

## Affiliations

**Sonika Rani Narang**[1] · **Manish Kumar Jindal**[2] · **Munish Kumar**[3]

[1] Department of Computer Science, D.A.V. College, Abohar, Punjab, India

[2] Department of Computer Science & Applications, Panjab University Regional Centre, Muktsar, Punjab, India

[3] Department of Computational Sciences, Maharaja Ranjit Singh Punjab Technical University, Bathinda, Punjab, India