Character Segmentation of Hindi Unconstrained Handwritten Words

Soumen Bag[1(✉)] and Ankit Krishna[2]

[1] Department of Computer Science and Engineering, ISM Dhanbad, Dhanbad, India
bagsoumen@gmail.com
[2] Department of Computer Science and Engineering, IIIT Bhubaneswar, Bhubaneshwar, India
ankitkrishna.id@gmail.com

Abstract. The proper character level segmentation of printed or handwritten text is an important preprocessing step for optical character recognition (OCR). It is noticed that the languages having cursive nature in writing make the segmentation problem much more complicated. Hindi is one of the well known language in India having this cursive nature in writing style. The main challenge in handwritten character segmentation is to handle the inherent variability in the writing style of different individuals. In this paper, we present an efficient character segmentation method for handwritten Hindi words. Segmentation is performed on the basis of some structural patterns observed in the writing style of this language. The proposed method can cope with high variations in writing style and skewed header lines as input. The method has been tested on our own database for both printed and handwritten words. The average success rate is 96.93 %. The method yields fairly good results for this database comparing with other existing methods. We foresee that the proposed character segmenattion technique can be used as a part of an OCR system for cursive handwritten Hindi language.

Keywords: Character segmentation · Handwritten word · Header line detection · Hindi language · Lower modifier · Upper modifier · Structural approach · OCR

1 Introduction

In last two decades several works have been done in the computer recognition of handwritten words. But few of us believe that a computer will ever be able to read humans' handwriting as good as human. Even so, it does not hurt to try to develop technology which can approach the recognition ability of humans. Solving the character segmentation problem is one of the keys to putting character recognition technology to practical use.

Character segmentation is an operation that seeks to decompose an image of a sequence of characters into subimages of individual symbols [7] as shown in

R.P. Barneva et al. (Eds.): IWCIA 2015, LNCS 9448, pp. 247–260, 2015.
DOI: 10.1007/978-3-319-26145-4_18

(Fig. 1). It is one of the decision processes in a system for optical character recognition (OCR). The performance of character segmentation techniques depend on the quality of the scanned document because due to poor quality scanning and ink bleeding, it generally happens that the neighboring characters in the scanned image touch each other. Character segmentation is a major challenge for such degraded documents [14].

A wide variety of line, word, and character segmentation methods for printed documents of Indian languages are reported in the literature [12]. But segmentation of cursive handwriting still remains one of the most challenging problems in the area of handwritten character recognition. Bangla and Hindi are the two very popular Indian languages having this cursive property in writing style. Some of the prior works on Bangla handwritten character segmentation include in [4,6,8,13,16]. But evidence of works on Hindi character segmentation are just a few in number [5,9]. In this paper, we present a novel technique for segmentation of unconstrained handwritten Hindi words which is highly efficient over the other existing methodologies in literature. The key features of our proposed method are summarized as follows:

- Extensive use of the structural properties of characters in the segmentation process.
- Efficient to handle inputs with highly skewed header lines.
- Covers many different handwriting styles written by different individuals and gives correct output for them.

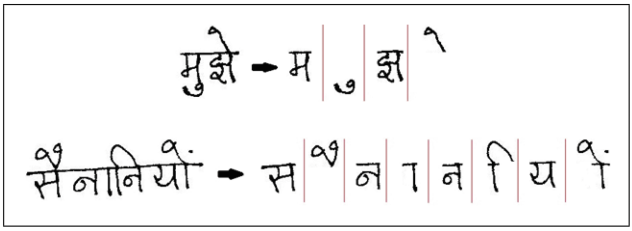


Fig. 1. Segmentation of handwritten Hindi words.

The rest of the paper is organized as follows. Section 2 describes the characteristics of Hindi language. Section 3 presents the proposed methodology for accurate segmentation of handwritten Hindi words. Experimental results and related discussions are reported in Sect. 4. The concluding notes are given in Sect. 5.

2 Properties of Hindi Language

Devanagari is the script for writing Hindi language. It consists of 14 vowels and 33 consonants. A sample set of basic Hindi alphabets is shown in Fig. 2(a).

The writing style is from left to right. There is no concept of lower or upper case alphabets as in English language [2]. Half form characters in Hindi increases the language complexity for recognition. The half characters may touch with full characters to make the characters called conjuncts or compound (see Fig.2(b)). In each conjunct character, the right part is a full consonant, and the left part is always a half consonant. When two or more characters are combined to form a word, the horizontal lines touch each other and generate a header line called *shirorekha*. The vowels modifiers can be placed at the left, right (or both), top, or bottom of the consonant. The vowels above the header line are called *ascenders* or upper modifiers and vowels below the consonants are called *descenders* or lower modifiers.

अ	आ	इ	ई	उ	ऊ	ऋ	ए	ऐ	ओ
औ	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						

(a) Sample images of Hindi Basic characters.

(b) Sample images of Hindi conjunct characters.

Fig. 2. Hindi alphabet set.

3 Proposed Methodology

The proposed method has three phases. A preliminary segmentation process extracts the header line and delineates the upper-strip from the rest in phase 1. This yields vertically separated middle zone and bottom zone components that may be conjuncts, touching characters, characters with lower modifier attached to it, shadow characters, or a combination of these. In phase 2, statistical information about these intermediate individual components is collected and the segmentation of upper modifier is performed. In phase 3, this statistical information

is used again to select the components on which further segmentation needs to be attempted. This separates the lower modifiers from the middle zone components. This segmentation methodology is performed in the following hierarchical order as shown in Fig. 3.

1. Scan the handwritten Hindi words needed to be segmented and perform the binarization using an adaptive-cum-interpolative method as described in [3]. It is noticed that binarization plays an important role in character segmentation. In this method, a multiscale framework is added to adaptive version of Otsu's method [11] to handle noises in different scales. To convert Otsus method to an adaptive model, instead of computing the global threshold value for the whole image, it computes the local threshold value for each pixel by observing the intensity behavior of its neighbor pixels.
2. Detect the header line and remove it completely.
3. Segment the upper modifiers left in the upper zone, if any, and make appropriate joining if required.
4. Identify the middle zone components containing any lower modifiers and segment these lower modifiers if required.
5. Finally, the segmented result is presented for further recognition process.

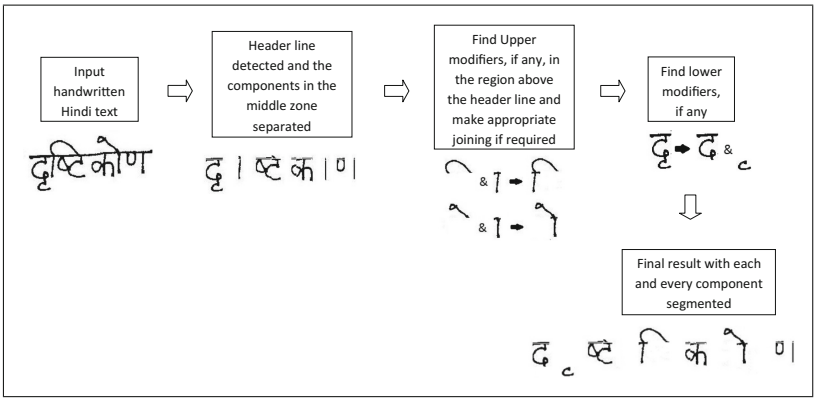


Fig. 3. System architecture of the proposed method.

3.1 Detection and Removal of Header Lines

Figure 4 outlines the proposed method for detecting and removing the header lines even if they are skewed in nature. The following steps discuss the method in detail.

1. Perform thinning of binarized handwritten Hindi words to get single pixel thin skeletons using Huang's method [10].
2. Find the start row, end row, start column, and end column for the span of a word.

3. Get the horizontal density of number of object pixels for each row in the upper half of the word height, i.e., from 'start row' to '(start row+end row)/2'.
4. Find the highest density row (marked as 'record') from the above list and consider to be the approximate header line row.
5. Divide the entire word width into stripes. The number of stripes is equal to ((2×width)/lower height) of the input word, where width = (end column-start column) and lower height = (end row-record).
6. Find the row having highest density of object pixel for each stripe by scanning from 'start row' to 'record'+7. This threshold value is set as per the experimental analysis.
7. Find the difference between 'record' and the local maximum row (from step 6) for each stripe and accordingly shift the entire stripe upwards or downwards based on the sign of the difference.

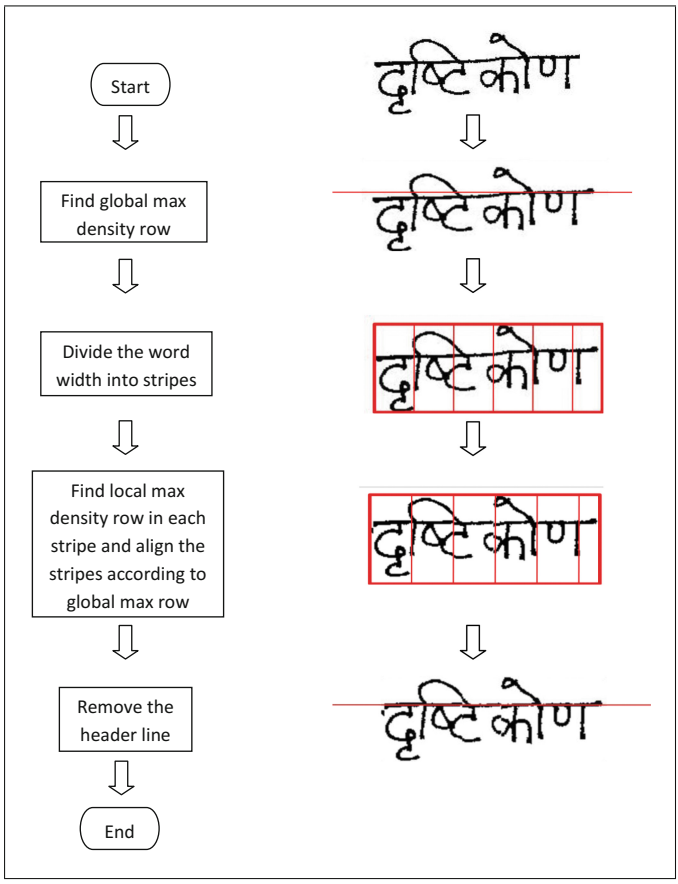


Fig. 4. Flowchart for header line detection and removal.

8. Finally, remove the row 'record' and appropriate number of rows above and below of it (according to the pen width of the input word) to get rid of the header line.

After removal of header line, we calculate the vertical density of object pixels for each column. We identify the columns having zero count for the above vertical density and use them as breakdown columns to separate the components at these positions.

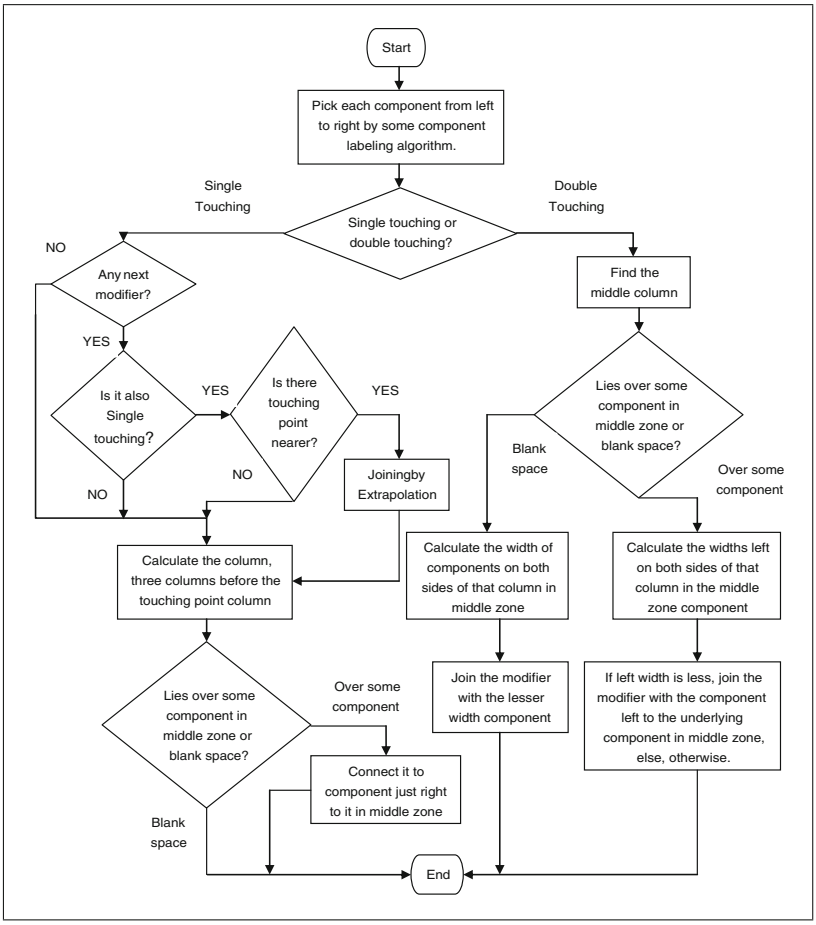


Fig. 5. Flowchart for upper modifier segmentation.

3.2 Segmentation of Upper Modifiers

To deal with upper modifiers separately, we extract it out one by one using Rosenfeld and Kak component labeling algorithm [15]. Figure 5 outlines the proposed technique for upper modifier segmentation. The steps are as follows:

1. The upper modifiers in Hindi language can be classified into two classes. This classification is done based on the number of times they touch the header line at different positions as shown in Fig. 6. They touch either once ( ) or twice ( ) as per the characteristics of writing style.
2. If it is a single touch, then check the class of the next modifier (if there is any) in the sequence from left to right. If there is no next modifier or the next modifier is not a single touching modifier then go to step 3. But if it is a single touching modifier, then check whether the touching points of present and next modifiers are nearer to each other or not. If they are not, then go to step 3. Otherwise, join them by applying the extrapolation method used in [1] and then move to the next step. All different possibilities of upper modifier are shown in Fig. 7.
3. Find the column which is four columns preceding from the touching point (see Fig. 8). If this column lies over blank space in middle zone then connect it to the component in the middle zone which is just right to the above calculated column. If it lies over some component in the middle zone then simply represent it separately.
4. If it is double touching, then find the middle column for the column span of the upper modifier. In Fig. 9, the middle column is marked by red line and the column span of the upper modifier is marked by green lines. If the middle column lies over some component in the middle zone (see Fig. 9(a)), then calculate the width on left (W_l) and right (W_r) sides of that column in the middle zone of that component. If W_l is less than W_r then join the modifier with the component in the middle zone which is just left to the underlying component in the middle zone. Otherwise, join it with the component in the middle zone which is just right to the underlying component in the middle zone.
5. If the middle column lies over blank space in the middle zone as shown in Fig. 9(b), then calculate the width of the components on both sides (C_l and C_r) of that column in the middle zone. If C_l is less than C_r then join the modifier with the component of width C_l; otherwise, join the modifier with the component of width C_r.

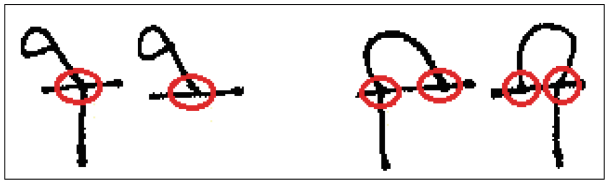
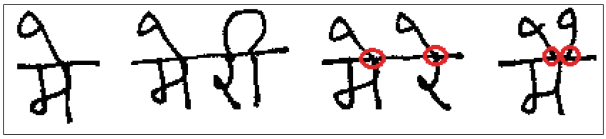
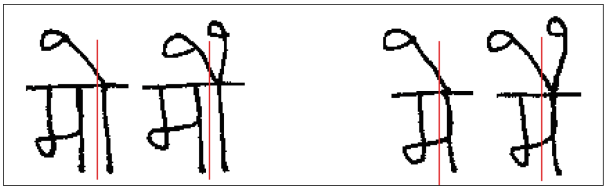


Fig. 6. (a) Single touching upper modifiers. (b) Double touching upper modifiers. Red circle indicates the touching point (Color figure online).



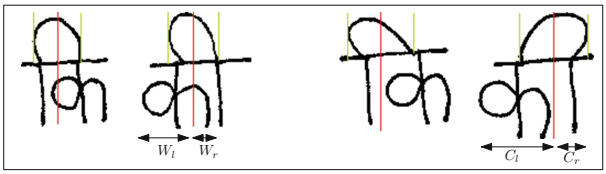
(a) (b) (c) (d)

Fig. 7. (a) No next modifier. (b) Next modifier is not a single touching modifier. (c) Next modifier is single touching but touching points are not nearer to each other. (d) Next modifier is single touching and touching points are nearer to each other. Red circle indicates the touching point (Color figure online).



(a) (b)

Fig. 8. Calculated column (marked in red) lies over (a) blank space in middle zone; (b) some component in middle zone (Color figure online).



(a) (b)

Fig. 9. Middle column (in red) lies over (a) some component in middle zone; (b) blank space in middle zone (Color figure online).

3.3 Segmentation of Lower Modifiers

The proposed method for lower modifier segmentation are shown in Fig. 11. At first, the components containing the lower modifiers are identified. Thereafter, they are classified into three classes as given below.

- Middle bar characters (क फ)
- Right bar characters (ख ग घ च ज झ त थ ध न प ब भ म य ल व स)
- No bar characters (छ द र ह)

1	2	3
4	5	6
7	8	9

Fig. 10. 3 × 3 grid for lower modifier detection.

The steps to determine the classes are:

1. Divide the entire component into 3×3 grid as shown in Fig. 10.
2. Consider block 3 and 6 and check whether more than 90 % of their rows contain object pixels or not. If so, then the component is a right bar character.
3. Consider block 2 and 5 and perform similar check over their rows. If it is satisfied, then it is a middle bar character.
4. If none of the cases are satisfied, then it is a no bar character.

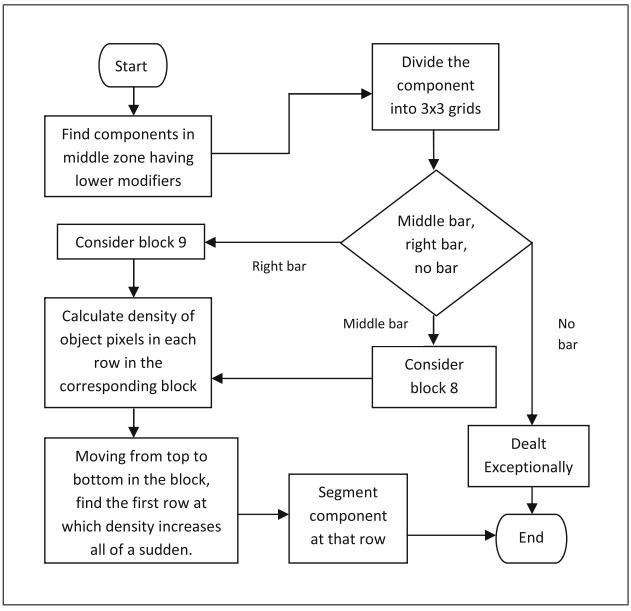


Fig. 11. Flowchart for lower modifier segmentation.

After the final classification, if it is a right bar character then block 9 is considered and if it is a middle bar character then block 8 is considered in the grid. Thereafter, the density of object pixels in each row in the considered block is

calculated. Then, moving from top to bottom in that block, the first row at which the density increases all of a sudden is found. This row is omitted to segment the lower modifier from its middle zone component. The no bar characters can be dealt exceptionally.

4 Experimental Results and Discussion

This section presents the experimental results and related discussion of our proposed method.

4.1 Experimental Dataset

Our dataset consists of about 12750 Hindi word samples among which 1200 samples are printed and 11550 samples are handwritten. The handwritten samples are collected from 30 different writers used 10 different types of pens with varying pen width. The samples are scanned at 300 dpi using HP Office Jet 5610 scanner. The implementation has been done on MATLAB (R2010a).

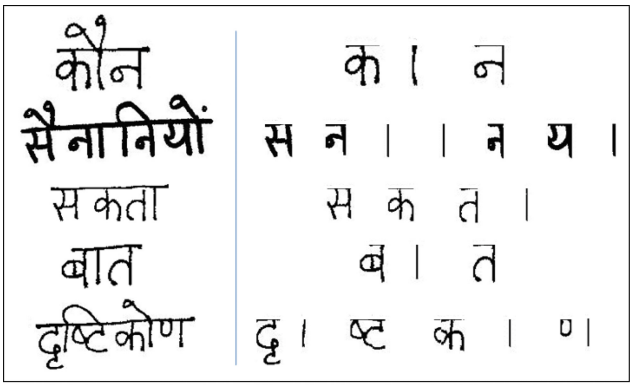
4.2 Character Segmentation Results

Detection of Header Lines: The experimental results are shown in Fig. 12(a). In this figure, the left and right column shows the input samples and the corresponding outputs after the completion of phase 1. It is shown that there is a white single width straight line detected as the header line for each of the inputs. This represents the required row to be removed. Now, we can remove the appropriate number of rows above and below of the obtained row according to the pen width of the written text so as to completely get rid of the header line.

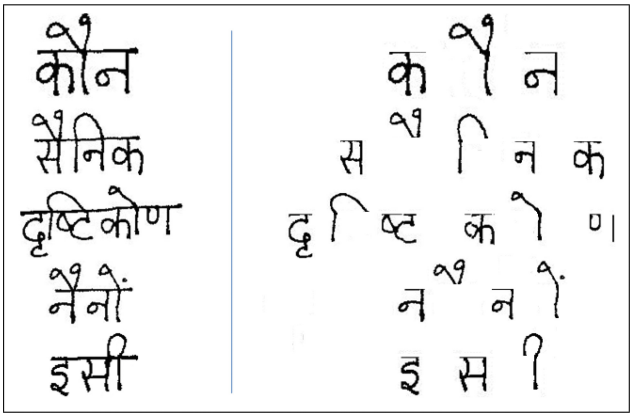
Segmentation of Upper Modifiers: The experimental results are shown in Fig. 12(b). The left and right column in the figure shows the inputs and the corresponding outputs after the completion of phase 2. We can observe that all the upper modifiers along with their middle zone counterpart get totally separated from the rest and are represented individually. For the upper modifiers with their counterparts in the middle zone, appropriate joining has been done and also the extrapolation has been performed if required.

Segmentation of Lower Modifiers: Finally, the lower modifiers are segmented in the last phase of the proposed method. The experimental results are shown in Fig. 12(c). The left and right column in the figure shows the inputs and the corresponding outputs after the completion of third phase. It is observed that all the lower modifiers are detected and segmented from their middle zone component character correctly.

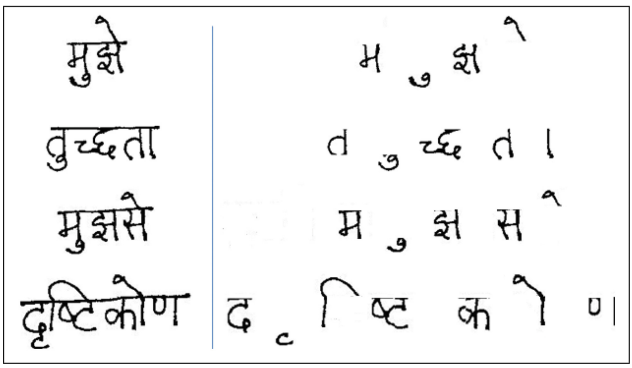
For the above all test cases, we consider the input samples having the shortcomings mentioned earlier to show the efficacy of our proposed method.



(a) Header line detection and removal.



(b) Segmentation of upper modifiers.



(c) Segmentation of lower modifiers.

Fig. 12. Experimental results of different phases for handwritten Hindi words.

Table 1. Header line detection accuracy.

Method	Total no. of words	No. of header lines detected correctly	Accuracy
Hanmandlu and Agrawal	12750	6739	52.85 %
	1200 (printed)	1168	97.33 %
	11550 (handwritten)	5571	48.23 %
Bansal and Sinha	12750	4022	31.55 %
	1200 (printed)	1168	94.85 %
	11550 (handwritten)	2854	24.7 %
Proposed	12750	12614	98.93 %
	1200 (printed)	1200	100 %
	11550 (handwritten)	11414	98.82 %

Table 2. Upper modifier segmentation accuracy.

Method	Total no. of words	Total no. of upper modifiers	No. of upper modifiers segmented correctly	Accuracy
Hanmandlu and Agrawal	12750	3693	2977	80.61 %
	1200 (printed)	704	640	90.9 %
	11550 (handwritten)	2989	2337	78.18 %
Bansal and Sinha	12750	3693	1648	44.62 %
	1200 (printed)	704	520	73.86 %
	11550 (handwritten)	2989	1128	37.73 %
Proposed	12750	3693	3558	96.34 %
	1200 (printed)	704	704	100 %
	11550 (handwritten)	2984	2854	95.45 %

4.3 Comparison with Other Methods

We compared our results with two existing methods of Bansal–Sinha [5] and Hanmandlu–Agrawal [9]. As the datasets used in these two methods were not available, so to perform the comparative analysis in same platform we prepared our own dataset with reasonable size as discussed in Sect. 4.1. We implemented the other two methods and tested all these methods on our own dataset. Our main objective was to make this comparative analysis unbiased. The accuracy of header line detection, upper modifier segmentation, and lower modifier segmentation are shown in Tables 1, 2, and 3 respectively. We observe that the performance of header line detection is much better than the other two existing methods. This is because of the efficiency of our proposed method to handle large variety of writing styles and skewed header lines as input data. Also for upper and lower modifier segmentation, our algorithm has shown an acceptable improvement in accuracy over the other two existing methods. The overall success rate of our proposed method is 96.93 % which is much better than Bansal–Sinha (48.58 %) and Hanmandlu–Agrawal (72.8 %) methods. During the measuring of accuracy rate, we treated over and under segmentation as an incorrect

Table 3. Lower modifier segmentation accuracy.

Method	Total no. of words	Total no. of lower modifiers	No. of lower modifiers segmented correctly	Accuracy
Hanmandlu and Agrawal	12750	3080	2616	84.94 %
	1200 (printed)	376	320	85.1 %
	11550 (handwritten)	2704	2296	84.91 %
Bansal and Sinha	12750	3080	2143	69.58 %
	1200 (printed)	376	336	89.36 %
	11550 (handwritten)	2704	1807	66.83 %
Proposed	12750	3080	2942	95.52 %
	1200 (printed)	376	360	95.74 %
	11550 (handwritten)	2704	2582	95.49 %

segmentation. This work can be extended later on to make a more generalized method for lower modifier segmentation in case of no bar characters and the segmentation of two characters touch in upper, middle, or lower region. Many a times shadow characters also occur in handwritten text when one totally independent component occurs under some other component. They can also be dealt with in future.

We have also done a comparison in between the above said methods and our proposed method w.r.t. computational time. We used our own test datasets for this experimental analysis. It is noticed that our proposed method is computationally efficient than the other two methods for both the printed and handwritten images.

5 Concluding Remarks

In this paper, we have proposed a character segmentation method based on structure shape of Hindi language. The proposed method has performed significantly well at each level of segmentation to handle large scale shape variation in writing style of Hindi language. The proposed method is tested on handwritten Hindi word images and the results are very promising with an average accuracy rate of 96.93 %. But this method is not performing well for few particular cases as stated earlier. In future, we shall extend our work to improve the accuracy of segmenattion and to make it applicable to character recognition for handwritten Hindi OCR system.

References

1. Bag, S., Harit, G.: Skeletonizing character images using a modified medial axis-based strategy. Int. J. Pattern Recognit. Artif. Intell. **25**, 1035–1054 (2011)
2. Bag, S., Harit, G.: A survey on optical character recognition for Bangla and Devanagari scripts. Sadhana **38**, 133–168 (2013)

3. Bag, S., Bhowmick, P., Behera, P., Harit, G.: Robust binarization of degraded documents using adaptive-cum-interpolative thresholding in a multi-scale framework. In: International Conference on Image Information Processing, pp. 1–6. IEEE Press, New York (2011)
4. Bag, S., Bhowmick, P., Harit, G., Biswas, A.: Character segmentation of handwritten Bangla text by vertex characterization of isothetic covers. In: National Conference on Computer Vision, Pattern Recognition, Image Processing and Graphics, pp. 21–24. IEEE Press, New York (2011)
5. Bansal, V., Sinha, R.M.K.: Segmentation of touching and fused Devanagari characters. Pattern Recognit. **35**, 875–893 (2002)
6. Bishnu, A., Chaudhuri, B.B.: Segmentation of Bangla handwritten text into characters by recursive contour Following. In: International Conference on Document Analysis and Recognition, pp. 236–239. IEEE Press, New York (1999)
7. Casey, R.G., Lecolinet, E.: A survey of methods and strategies in character segmentation. IEEE Trans. Pattern Anal. Mach. Intell. **18**, 690–706 (1996)
8. Garain, U., Chaudhuri, B.B.: Segmentation of touching characters in printed Devnagari and Bangla scripts using fuzzy multifactorial analysis. IEEE Trans. Syst. Man Cybern. Part C **32**, 449–459 (2002)
9. Hanmandlu, M., Agrawal, P.: A structural approach for segmentation of handwritten Hindi text. In: International Conference on Cognition and Recognition, pp. 589–597 (2005)
10. Huang, L., Wan, G., Liu, C.: An improved parallel thinning algorithm. In: International Conference on Document Analysis and Recognition, pp. 780–783. IEEE Press, New York (2003)
11. Otsu, N.: A threshold selection method from gray-level histogram. IEEE Trans. Syst. Man Cybern. **9**, 62–66 (1979)
12. Pal, U., Chaudhuri, B.B.: Indian script character recognition: a survey. Pattern Recognit. **37**, 1887–1899 (2004)
13. Pal, U., Datta, S.: Segmentation of Bangla unconstrained handwritten text. In: International Conference on Document Analysis and Recognition, pp. 1128–1132. IEEE Press, New York (2003)
14. Pal, U., Jayadevan, R., Sharma, N.: Handwritten recognition in Indian regional scripts: a survey. ACM Trans. Asian Lang. Inf. Process. **11**(1), 1–35 (2012)
15. Rosenfeld, A., Kak, A.C.: Digital Picture Processing, 2nd edn., vols. 1 and 2. Academic Press, New York (1982)
16. Sarkar, R., Das, N., Basu, S., Kundu, M., Nasipuri, M., Basu, D.K.: A two-stage approach for segmentation of handwritten Bangla word images. In: International Conference on Frontiers in Handwriting Recognition, pp. 403–408. CENPARMI, Canada (2008)