International Journal of Computational Intelligence Research.
ISSN 0973-1873 Vol.3, No.4 (2007), pp. 277–286

Segmentation of Horizontally Overlapping Lines in Printed Indian Scripts

M.K. Jindal[1], R.K. Sharma[2] and G.S. Lehal[3]

[1]Department of Computer Applications,
Panjab University Regional Centre,
Muktsar, Punjab, India
manishphd@rediffmail.com
[2]School of Mathematics and Computer Applications,
Thapar Institute of Engineering and Technology,
Patiala, Punjab, India
rksharma@tiet.ac.in
[3]Department of Computer Science,
Punjabi University, Patiala, Punjab, India.
gslehal@gmail.com

Abstract: **Horizontally overlapping lines are normally found in printed newspapers of any Indian script. Along with these overlapping lines few other broken components of a line (strip) having text less than a complete line are also found in text. The horizontally overlapping lines and other strips make it very difficult to estimate the boundary of a line leading to incorrect line segmentation. Incorrect line segmentation decreases the recognition accuracy. In this paper we have proposed a solution for segmenting horizontally overlapping lines and solved the problem of other strips in eight most widely used printed Indian scripts. Whole document has been divided into strips and proposed algorithm has been applied for segmenting horizontally overlapping lines and associating small strips to their respective lines. The algorithm has shown approximately 96.45-99.79% accuracy depending upon script. We have also tried to segment horizontally overlapping lines, containing different sized text, i.e. the newspaper articles in which bigger sized heading lines overlaps with normal sized text lines.**

Keywords: Horizontally overlapping lines, Strips, Indian scripts, Headline based scripts, Non headline based scripts.

I. Introduction

A lot of research work has been investigated for character recognition of Indian scripts. For an optical character recognition (OCR) system, segmentation phase is an important phase and accuracy of any OCR heavily depends upon segmentation phase. Incorrect segmentation leads to incorrect recognition. Segmentation phase includes line, word and character segmentation. Before word and character segmentation, line segmentation is performed to find the number of lines and boundaries of each line in any input document image. Incorrect line segmentation may result in decrease in recognition accuracy. The simplest and most widely used method to segment the lines is to use the inter-line gap in horizontal projection as line boundaries. This technique does not work well on many documents in Indian scripts. It results in strips containing multiple horizontally overlapping lines leading to under segmentation and strips containing components of a line leading to over segmentation. We have proposed an algorithm for segmenting horizontally overlapping lines in eight most widely used printed Indian scripts. These scripts are Gurmukhi, Devanagari, Bangla, Gujarati, Kannada, Tamil, Telugu and Malayalam. The problem of multiple horizontally overlapping lines is common in printed newspapers of these Indian scripts due to high compression methods used for printing of the newspapers.

We have not found much reported work for segmenting horizontally overlapping lines in Indian scripts. Bansal [1] has discussed a two-pass algorithm based upon average line height to solve the problem of horizontally overlapping lines in Devnagari script. Harikumar *et al.* [2] have used the concept of average line height to segment the horizontally overlapping lines in Malayalam script. Pal and Datta [3] have segmented unconstrained handwritten text lines by dividing the text into vertical strips and then taking horizontal projections. Pal *et al.* [4] have used various features of Indian scripts like existence of headline, number and position of peaks in horizontal projections, water reservoir etc. to separate various lines from multi script document. Pal and Chaudhuri [5] have used structural and statistical features for separating machine printed text lines from hand-written text lines for both Bangla and Devnagari scripts. Dholakia *et al.* [6] have used slopes of connected components to find the three zones in the printed Gujarati

script. They have assumed that lines have been segmented properly before finding the zones. This is not possible in case of horizontally overlapping lines. To the best of our knowledge, no author has discussed in detail the problem of over segmentation of lines in a document. As the problem of over segmentation has been discussed first time this is not any modification to the previous work. The fragments of the line produced due to over segmentation make the problem of segmentation of lines very hard. The methods discussed in literature [1, 2] for segmenting overlapping lines fail when there is over segmentation. Also the methods [1, 2] will not work that accurately when there are more than two overlapping lines.

Pal and Chaudhuri [7, 8] have also discussed the concept of zoning and line segmentation. Lehal (one of coauthor) and Singh [9-11] have discussed the simple methods of line, word and character segmentation and nothing has been discussed for segmentation of overlapping lines. Hence this is not any duplicate work.

In this paper, we have proposed new strategies to segment the horizontally overlapping lines and joining together the components of over segmented lines for eight printed Indian scripts. The samples have been taken form the newspapers of the corresponding scripts. Also the problem of segmenting overlapping lines of different sized text has been discussed.

The rest of the paper is organized as follows. In the next Section, characteristics of Indian scripts have been described briefly and Section III describes the preprocessing steps taken. In Section IV, the problem of horizontally overlapping lines has been defined and an algorithm has been proposed to solve the problem. In Section V, the problem of segmentation of overlapping lines of different sized text has been solved, proposing a modified algorithm. Finally, Section VI consists of results and discussions.

II. Characteristics of Indian Scripts

There are 23 official languages in India [12] namely Assamese, Bengali, Bodo, Dogri, English, Gujarati, Hindi, Kannada, Kashmiri, Konkani, Maithili, Malayalam, Manipuri, Marathi, Nepali, Oriya, Punjabi, Sanskrit, Santhali, Sindhi, Tamil, Telugu and Urdu. There are 13 different scripts Assamese, Bangla, Devnagari, Gujarati, Gurmukhi, Kannada, Kashmiri, Malayalam, Oriya, Roman, Tamil, Telugu and Urdu used for writing these languages. The concept of upper/lower case characters is not present in Indian language scripts except English language script. Indian scripts can be divided into two groups. First group having the concept of headline (a horizontal line at top of the characters), e.g. Devnagari, Bangla and Gurmukhi script alphabets have headline. Fig. 1 contains example words from Gurmukhi script and line number 1 is called upper line, line number 2 is start of headline, line number 3 is end of the headline, line number 4 is called base line and line number 5 is lower line in this figure. Second group contains scripts not having the concept of headline, e.g. Gujarati, Kannada, Malayalam, Oriya, Tamil, Telugu etc. Fig. 2 contains example words from Gujarati script and line number 1 is called upper line, line number 2 is mean line, line number 3 is base line and line number 4 is lower line in this figure. A text line of almost every script can be partitioned into three horizontal zones namely, upper zone, middle zone and lower zone except Urdu script. Fig. 1 and 2 contains the examples showing the concept of three zones.

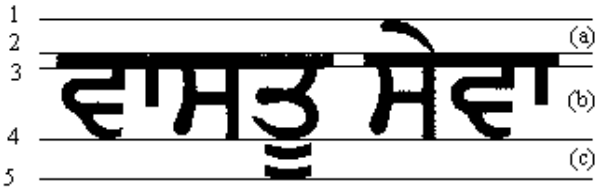

Figure 1. Gurmukhi script word: (a) upper zone from line number 1 to 2, (b) middle Zone from line no 3 to 4, (c) lower zone from line number 4 to 5.

Fig. 1(a), 1(b) and 1(c) show the contents of the three zones, i.e. upper, middle and lower zone respectively in Gurmukhi script (headline based script).

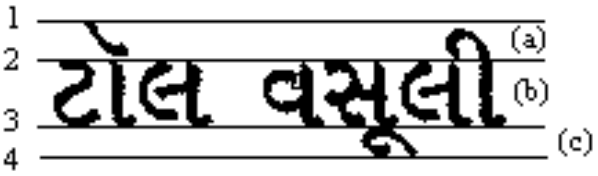

Figure 2. Gujarati script word: (a) upper zone from line number 1 to 2, (b) middle Zone from line no 2 to 3, (c) lower zone from line number 3 to 4.

Fig. 2(a), 2(b) and 2(c) show the contents of the three zones, i.e. upper, middle and lower zone respectively in Gujarati script (non headline based script).

III. Preprocessing

Preprocessing is applied on the input binary document in order to minimize the effect of spurious noise during segmentation of various strips and subsequently segmentation of various lines. In the present study, both salt and pepper noise have been removed using standard algorithm [13]. The skewness present in the document image has also been removed with the help of standard skew detection and removal algorithm [14]. The algorithm proposed in the present study does not perform well in case the image is skewed, because all the algorithms are based on the exact positions of the headline, baseline and meanline of each line in the input document. If the image is skewed it will not be possible to extract the exact position of headline, baseline or meanline and algorithm will not work that efficiently.

IV. Overlapping Lines Segmentation

Overlapping lines may exist in any script. Therefore segmentation of overlapping lines is very much required for text recognition purpose. Existence of other kinds of strips in printed Indian scripts makes the problem of line segmentation more difficult. In this section, we have developed an algorithm to segment the horizontally overlapping lines and joining together the components of over segmented lines. We have used the idea of horizontal projection and continuous vertical projection [1] in the algorithm. Also, a strip can be defined as a collection of consecutive run of horizontal rows containing at least one pixel.

It has been observed that in some particular documents such as newspapers of Indian scripts, line segmentation using horizontal projection method fails and results in either under segmentation or over segmentation. The lower zone characters of one line may touch with the upper zone characters of next line, thus producing multiple horizontally overlapping lines. Ideally horizontal projection should divide the document into horizontal strips, where each strip corresponds to a single line, but in Indian scripts it has been found that the strips contain components from other lines as broken parts of single line. We have identified 10 such kinds of strips as follows:

Type 1: strip containing only upper zone characters (strip number 1 in Fig. 3, strip number 2 in Fig. 9, strip number 3 in Fig. 17).

Type 2: strip containing only middle zone characters having upper zone and/or lower zone but this upper and/or lower zone has become part of some other strip (strips number 2 and 6 in Fig. 3).

Type 3: strip containing upper zone characters touching with middle zone characters having no lower zone characters, i.e. one line (strip number 3 in Fig. 3, strip number 2 in Fig. 5, strip number 1 in Fig. 9).

Type 4: strip containing upper zone characters touching with middle zone characters having lower zone characters, but the lower zone has been segmented into different strip (strip number 9 in Fig. 3, strip number 2 in Fig. 11, strip number 2 in Fig. 15).

Type 5: strip containing upper, middle and lower zone characters, i.e. complete one line (strip number 4 in Fig. 3, strip number 1 and 3 in Fig. 7, strip number 4 in Fig. 9, strips number 1, 4, 6 in Fig. 11, strip number 1 in Fig. 15, strips number 1, 5 in Fig. 17).

Type 6: strip containing upper, middle and lower zone characters of one line and upper zone of next line (strip number 5 in Fig. 3).

Type 7: strip containing only lower zone characters (strip number 7 in Fig. 3, strip number 3 in Fig. 11, strip number 6 in Fig. 15).

Type 8: strip containing lower zone characters touching with upper zone of next line (strip number 8 in Fig. 3, strip number 3 in Fig. 15).

Type 9: strip containing middle zone and lower zone characters whose upper zone exists but belongs to some different strip (strip number 3 in Fig. 9, strip number 4 in Fig. 17).

Type 10: strip containing two or more horizontally overlapping lines (strip number 8 in Fig. 3, strips number 3, 4 in Fig. 5, strip number 2 in Fig. 7, strip number 3 in Fig. 9, strip number 5 in Fig. 11, strips number 1, 2 in Fig. 13, strip number 4 in Fig. 15, strip number 2 in Fig. 17).

These kinds of strip lines make it very difficult to identify strip type. Also in case of multiple horizontally overlapping lines, it is difficult to estimate the exact position of pixel row, which segments one line from the next line. Statistical analysis of newspaper articles from various scripts reveals the information as shown in Table 1 and 2, about the percentage of occurrence of various kinds of strips.

Table 1. Percentage of various kinds of strip in Gurmukhi, Devanagari, Bangla and Gujarati scripts.

Script / **Strip type**	**Gur-mukhi**	**Deva-nagari**	**Bangla**	**Gujarati**
Type 1	0.65	0	0	13.34
Type 2	0.78	0	0	0
Type 3	10.04	49.56	12.65	0
Type 4	21.39	0	0	0
Type 5	12.98	8.1	83.23	40.00
Type 6	2.04	0	0	3.33
Type 7	19.35	0	0	0
Type 8	14.84	0	0	0
Type 9	0	0	0	3.33
Type 10	17.93	42.34	04.12	40.00

Table 2. Percentage of various kinds of strips in Kannada, Tamil, Telugu and Malayalam scripts.

Script / **Strip type**	**Kannada**	**Tamil**	**Telugu**	**Malaya-lam**
Type 1	0	8.69	0	2.70
Type 2	0	0	0	0
Type 3	9.67	0	8.72	0
Type 4	6.45	0	0	0
Type 5	61.31	30.43	65.21	72.98
Type 6	0	0	0	0
Type 7	9.67	0	4.34	0
Type 8	0	0	4.34	0
Type 9	0	8.69	0	0
Type 10	12.90	52.19	17.39	24.32

These results have been obtained by analyzing single column news articles from newspapers of corresponding script. We have taken sample of around 50-100 news items of each script depending upon availability of different newspapers in each script. Zero in any entry in Table 1 and Table 2 indicates that we have not identified that kind of strip out of available data (newspaper articles). But there remains a possibility of finding these kinds of strips also. It can be seen from the Table 1 and 2 that problem of horizontally overlapping lines in Devanagari and Tamil is acute and moderate in other scripts.

It may be noted that there are ten strips in Fig. 3, but actual number of lines are nine. Strips 3 and 4 constitute a complete line and need no segmentation. Strip 8 contains overlapping lines, which require proper segmentation. Strips 1 and 7 contain components from upper and lower zone and require the decision that these are part of which strip in order to make a complete line. Similarly strip number 2 and 6, which contains only middle zone, needs to include its upper and lower zones. Strip number 5 and 10 contains extra lower/ upper zone of some adjoining line. As such, it is necessary to find the exact boundaries of these lines.

Algorithm *seglines* has been developed for segmenting horizontally overlapping lines and joining together the components of over segmented lines. This algorithm segments the whole document into individual lines. Its input is single column news article of any script from eight scripts and its output is document with proper line boundaries.

Algorithm s*eglines*

BEGIN

Step 1: Using the horizontal projections, different strips in input binary document are identified. For that whenever $HP(i) = 0$ for $i = 1, 2, 3, \ldots, L$, it is marked as the boundary of strip line. Let us denote the strips by $S_1, S_2, S_3, \ldots, S_m$ and first row of strip p as $FR(S_p)$, last row of strip p as $LR(S_p)$. Height of the strip is calculated by $H(S_p)=LR(S_p)-FR(S_p) + 1$, for $p = 1, 2, 3, \ldots, m$. Strips identified in document from eight scripts are shown in Figs. 3, 5, 7, 9, 11, 13, 15 and 17.

Step 2: if input document is from any headline based script, go to step 3 else go to step 4.

Step 3: identify the position of headlines using horizontal projections. Denote the ending position of the headlines as $H_1, H_2, H_3, \ldots, H_n$. Also denote the lines to be identified as $L_1, L_2, L_3, \ldots, L_n$.

//number of headlines is same as number of actual //lines

Go to step 5.

Step 4: identify the position of meanlines, using first order differences of horizontal projections. Denote the position of the meanlines as $H_1, H_2, H_3, \ldots, H_n$.

//number of meanlines are same as number of actual //lines.

Step 5: define

$$AVG\_LINE\_HEIGHT = \frac{1}{n-1}\sum_{i=2}^{n}(H_i - H_{i-1})$$

Step 6: set $LINE\_NO = 1$ and first row of line $LINE\_NO$ as first row of first strip, i.e. $FR(L_{LINE\_NO}) = FR(S_1)$.

Step 7: for $i = 1$ to m, perform the following operations:

Step 7.1: if $H(S_i) <$ **P1 *** AVG_LINE_HEIGHT, S_i is of type 1.

//contains only upper zone

Repeat step 7.

//ignore current strip and go for next strip.

Step 7.2: if $H(S_i) > 0.50 * AVG\_LINE\_HEIGHT$, S_i will be of type 2, 3, 4, 5, 6, 8, 9 or 10 and will contain at least one headline/meanline and one baseline.

Step 7.3: identify the position of baseline. Mark it as $BASE_{LINE\_NO}$. Also set height of the middle zone as $HGT\_MID = BASE_{LINE\_NO} - H_{LINE\_NO}$.

Step 7.4: set last row of line $LINE\_NO$ as $LR(L_{LINE\_NO}) = BASE_{LINE\_NO} +$ **P2** $*(HGT\_MID)$.

//This will solve the segmentation problem of strip //type 2, 3, 4, 5, 6, 8 and 9.

Step 7.5: if $LR(S_i) > LR(L_{LINE\_NO})$

//strip type 10 containing horizontally overlapping //lines

set $H(S_i) = H(S_i)-(LR(L_{LINE\_NO})-FR(L_{LINE\_NO}))$, increment $LINE\_NO$. Also set $FR(L_{LINE\_NO}) = LR(L_{LINE\_NO-1}) + 1$ and go to step 7.1.

//for same strip.

Step 7.6: if $LR(S_{i+1}) <= LR(L_{LINE\_NO})$, increment i.

//strip type 7 containing only lower zone

Repeat step 7.6.

//for multiple lower zones.

Step 7.7: increment $LINE\_NO$. Set $FR(L_{LINE\_NO}) = LR(L_{LINE\_NO-1}) + 1$. Go to step 7.

//for next strip.

Step 8: for $j = 1$ to $LINE\_NO$

Display $FR(L_j)$ to $LR(L_j)$ as line boundaries.

END.

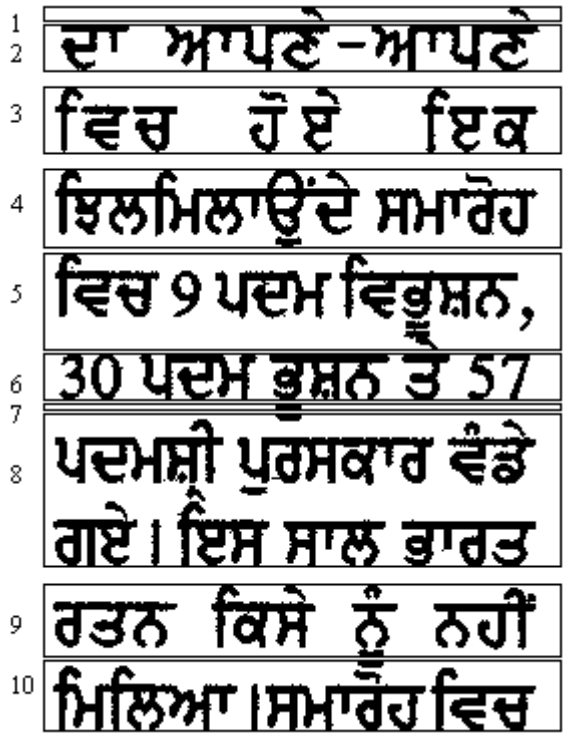

Figure 3. Various strips in printed Gurmukhi script.

Figure 4. Different line boundaries identified of Fig. 3 using proposed algorithm.

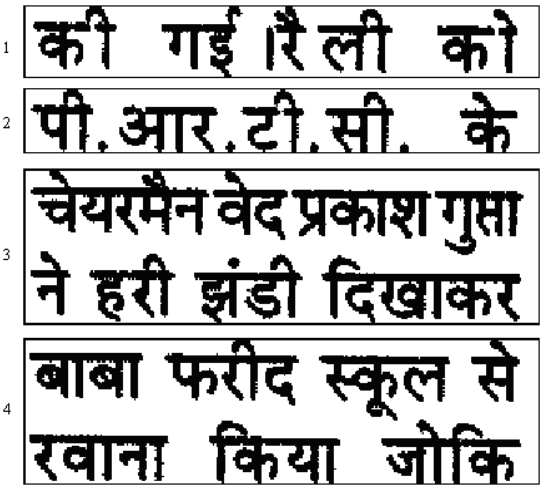

Figure 5. Various strips in printed Devanagari script.

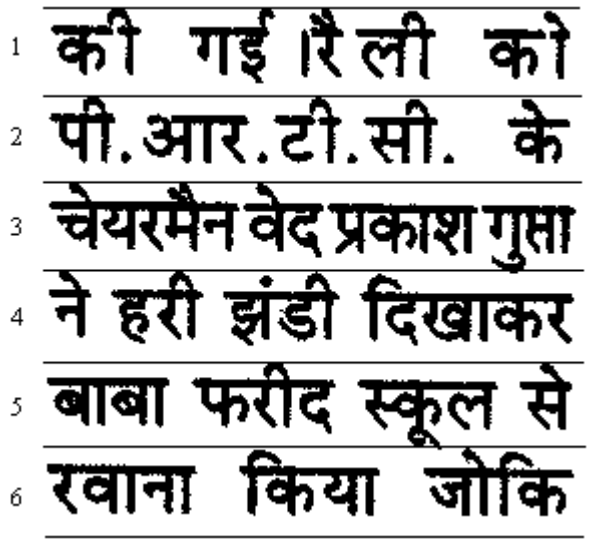

Figure 6. Different line boundaries identified of Fig. 5 using proposed algorithm.

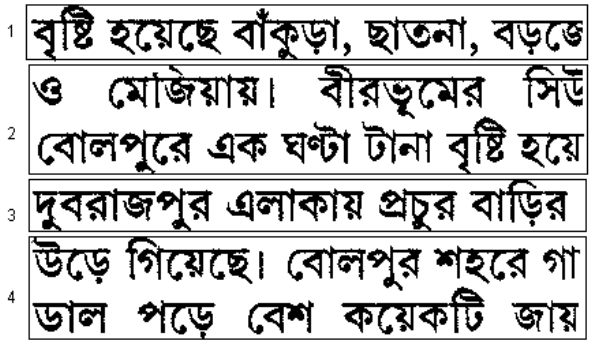

Figure 7. Various strips in printed Bangla script.

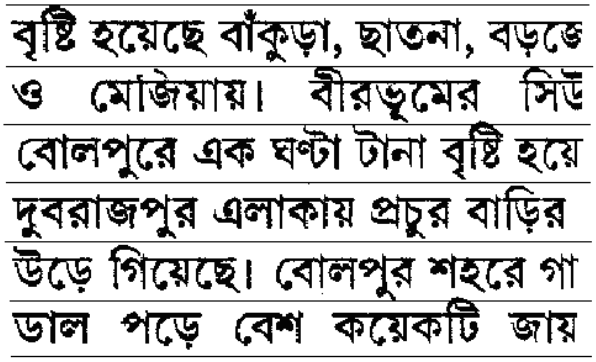

Figure 8. Different line boundaries identified of Fig. 7 using proposed algorithm.

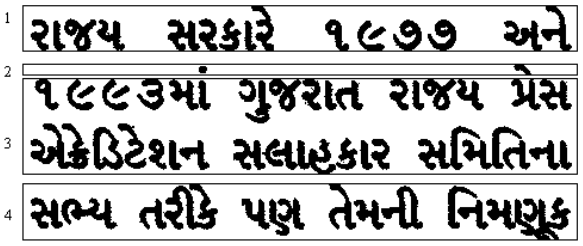

Figure 9. Various strips in printed Gujarati script.

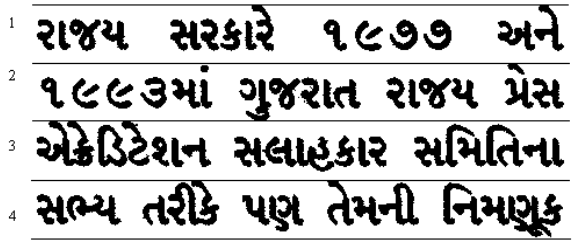

Figure 10. Different line boundaries identified of Fig. 9 using proposed algorithm.

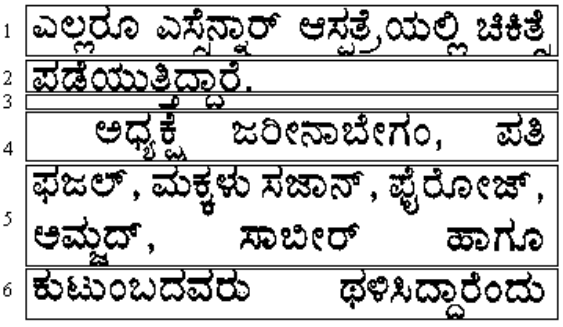

Figure 11. Various strips in printed Kannada script.

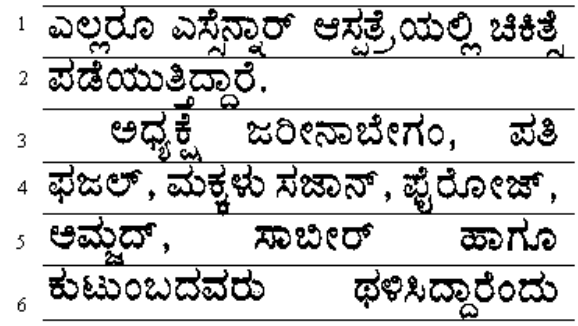

Figure 12. Different line boundaries identified of Fig. 11 using proposed algorithm.

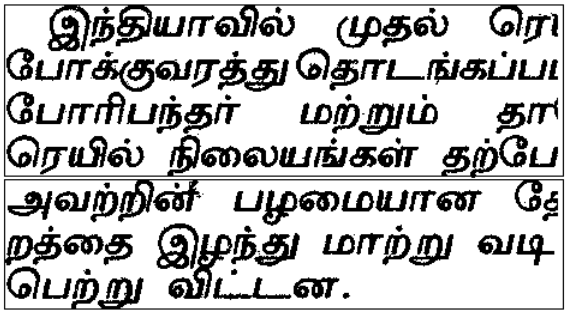

Figure 13. Various strips in printed Tamil script.

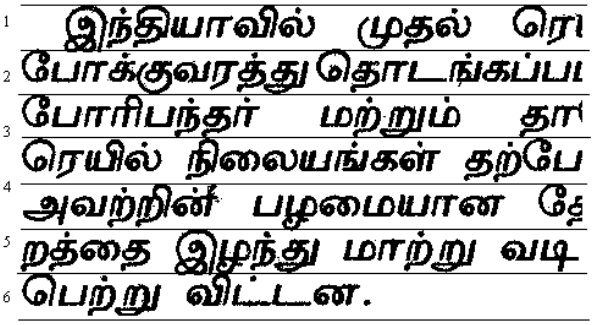

Figure 14. Different line boundaries identified of Fig. 13 using proposed algorithm.

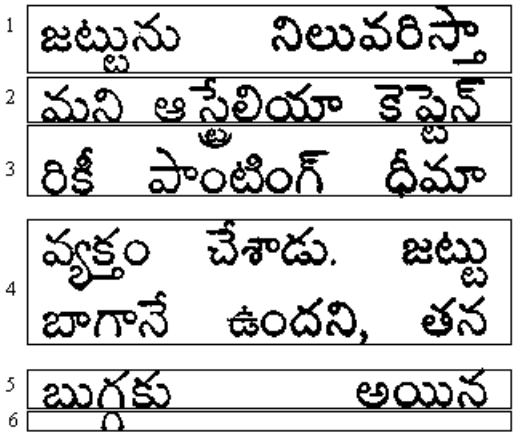

Figure 15. Various strips in printed Telugu script.

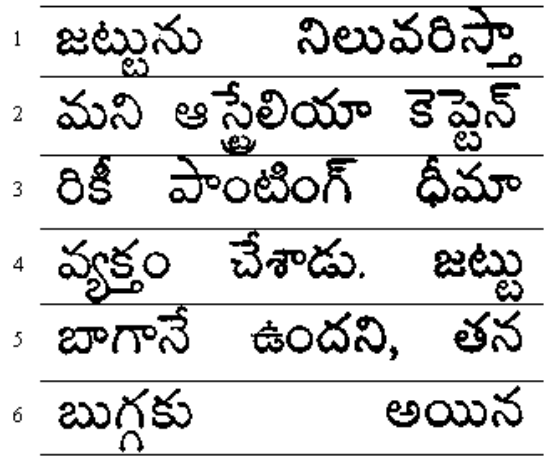

Figure 16. Different line boundaries identified of Fig. 15 using proposed algorithm.

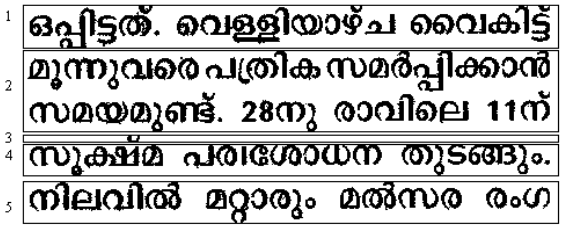

Figure 17. Various strips in printed Malayalam script.

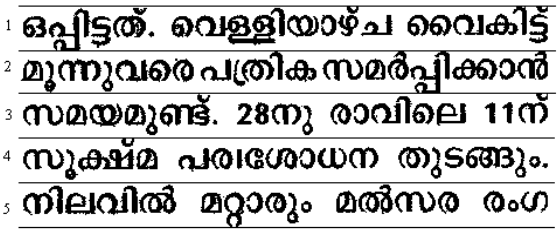

Figure 18. Different line boundaries identified of Fig. 17 using proposed algorithm.

Figs. 4, 6, 8, 10, 12, 14, 16, 18 show the boundaries of different lines identified for different scripts using the proposed algorithm. This algorithm is developed on the basis of a heuristic that makes a relation between the height of middle zone with height of lower zone and upper zone. We have chosen two parameters **P1** and **P2**. **P1** parameter has been defined as the ratio of the height of upper zone with average line height (*AVG_LINE_HEIGHT*) and **P2** defines the height of the lower zone with respect to middle zone. On the basis of experimental analysis on various news articles of various scripts, we have adjusted the value of **P1** and **P2**, e.g. for all headlines based scripts the height of the middle zone is approximately equal to the double the height of lower zone, so **P2** is set as 0.50 (height of lower zone/height of middle zone) for headline based scripts. Similarly based upon the value of height of lower zone/height of middle zone, the value of **P2** is set for different scripts.

Value of **P1** and **P2** differs for different scripts due to the structural properties of the lower and upper zone characters of these scripts. We have shown the values of **P1** and **P2** for different scripts in Table 3.

Table 3. Value of **P1** and **P2** for different scripts.

Script	Value of P1	Value of P2
Gurmukhi	0.30	0.50
Devnagari	0.30	0.50
Bangla	0.30	0.50
Gujarati	0.30	0.50
Kannada	0.25	0.70
Tamil	0.30	0.70
Telugu	0.40	0.80
Malayalam	0.30	0.70

The values of **P1** and **P2** have been set in accordance with the available sample data. These values can be adjusted in accordance with the different kinds of samples available in different scripts.

For finding the position of headline in algorithm *seglines*, for headline based scripts, one can use any standard technique [1, 7, 8, 11] and for non headline based script standard technique [6] can be used. We have used the following code for identification of headline.

Step headline:

if script= headline based
    find *MAXPIX* = max{*HP(i)*}, *i* = 1, 2, 3, …,*L*. The headlines are considered as those lines whose *HP(i)* ≥ 70% of *MAXPIX*
else
    find first order differences of horizontal projections, i.e. *dx*[*i*]=*HP*[*i*+1]-*HP*[*i*].
    if script=Gujarati
        while *dx*[*i*]>0 increment *i;*
        start=i
        while *dx*[i]<=0 increment *i;*
        end=i
        find max(*HP*[*j*]) for *j=end* to (*end*+(*end-start*))
        row number *j* is meanline called *mean*.
        upper=mean-start
    else
        find *maxh*=max(*HP*[ ]) for *i* = 1, 2, 3, …,*L*.
        while *dx*[*i*]<0.55**maxh* increment *i*;
        i marks the meanline.

For finding the position of baseline, for headline based scripts, one can use any standard technique [1, 7, 8, 11]. We have used the following code identification of baseline for both headline based and non headline based scripts.

Step baseline:
If script=headline based
identify the position of baseline by noting the continuous vertical projection *CVP(k)*, {$k=H_{LINE\_NO}$ to $LR(S_i)$}. The position where *CVP(k)* ends, mark it as α, every time. The row in which maximum α are found is considered to be the baseline.
else
find first order differences of horizontal projections, i.e. *dx*[*i*] =*HP* [*i*+1]-*HP*[*i*].
if script=Gujarati
Find min(*HP*[*j*]), for *j*= *mean* + *upper* * 1.5 to *mean* + *upper* * 2.5.
row number *j* is baseline.
else
find *minh*=min(*HP*[ ]) for *i* = 1, 2, 3, …,*L.*
while *dx*[*i*]>0.55 * *minh* increment *i*;
i marks the baseline.

The algorithm *seglines* has shown a remarkable improvement in accuracy, for segmenting the horizontally overlapping lines and associating the broken components of a line to their respective lines. This algorithm works even if the input document contains many consecutive horizontally overlapping lines. As shown in Fig. 13, there are four consecutive horizontally overlapping lines in strip number 1 and three consecutive horizontally overlapping lines in strip number 2. The proposed method segments all the lines of these strips correctly into individual lines.

These kinds of problems have been found in newspapers of almost all printed Indian scripts. We have analyzed and solved the problem in eight most widely used printed Indian scripts namely Gurmukhi, Devnagari, Bangla, Gujarati, Kannada, Tamil, Telugu and Malayalam. The same algorithm can be used for segmenting overlapping lines in Oriya script.

The algorithm *seglines* fails in case of Urdu script. As shown in Fig. 19 there are horizontally overlapping lines in Urdu script also. Strip number 1 contains two and strip number 2 contains four horizontally overlapping lines in Fig. 19. There is no concept of zoning in Urdu script and algorithm *seglines* is based upon upper zone, middle zone and lower zone, hence algorithm will not segment horizontally overlapping lines in this case.

IV. Overlapping Lines Segmentation of Different Text Size.

Newspapers contain the text with a large variation in text size. The headlines of every news is roughly always larger in size than the actual news text. One can infer from Fig. 20 that first two lines that are heading of the news have larger text size (let us call it segment 1) than the text size of the news text (let us call it segment 2). The algorithm *seglines* works accurately for segmenting overlapping lines of same text size, but when two different texts sized lines overlaps the algorithm does not work that accurately. Fig. 21 contains the output of the algorithm *seglines* when applied on the text given in Fig. 20. One can see that line number 2 and 3 are not properly segmented. Line number 2 has overlapped with line number 3, thus producing incorrect segmentation by breaking some text portion of the third line and adding it to the second line. Other lines have been correctly segmented. As such, two consecutive lines having different text size are not properly segmented.

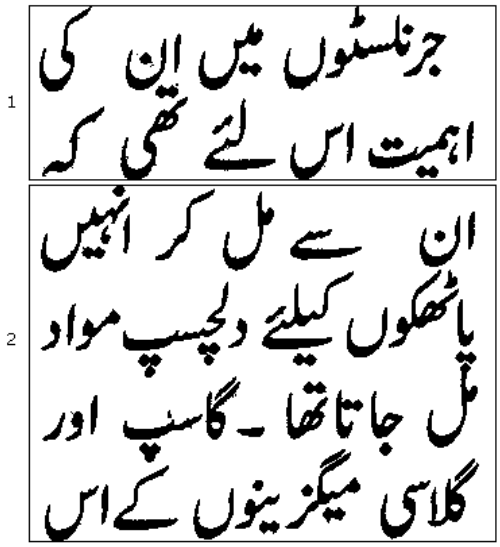

Figure 19. Urdu script containing overlapping lines.

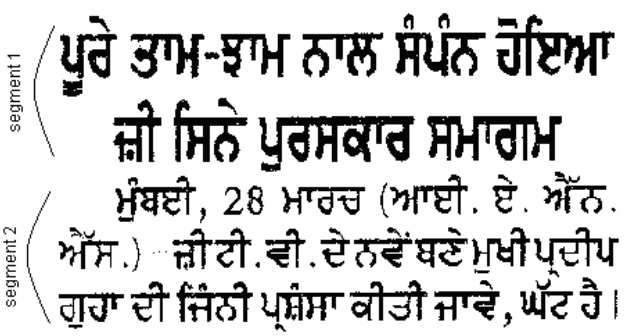

Figure. 20. Printed text of Gurmukhi script containing mixed font size characters.

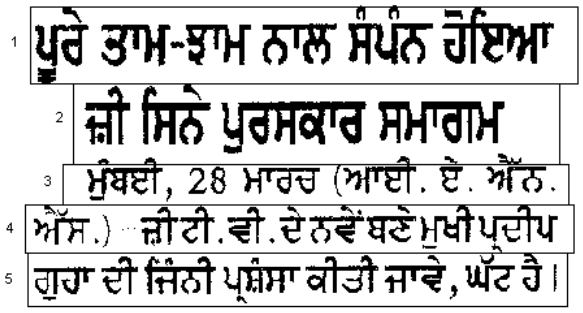

Figure 21. Different line boundaries of Fig. 20 identified using proposed algorithm.

We have modified the algorithm *seglines* in order to solve the problem of different text sizes by considering the fact that the number of lines in segment 1 is less than the number of lines in segment 2 for almost all news items. As such, *AVG_LINE_HEIGHT* will be closer to average line height of segment 2. Modified step 7 of the algorithm *seglines* is given below:

Step 7: for i=1 to m perform the following operations:

Step 7.1: //for strip from segment 1
if (H_{LINE_No+1}- H_{LINE_No}) > 1.4 * (*AVG_LINE_HEIGHT*))
PT=0.35,*PM*=0.60 goto step 7.4.

Step 7.2: //for strip joining segment 1 and segment 2
if ((H_{LINE_No+1}- H_{LINE_No}) > 1.15 * (*AVG_LINE_HEIGHT*))
PT=0.35, *PM*=0.60, *flag*=1, go to step 7.4.

Step 7.3: //for strip from segment 2
if ((H_{LINE_No+1}- H_{LINE_No}) < (*AVG_LINE_HEIGHT*)) set *flag1*=1 and *PT*=0.25, *PM*=0.40.

Step 7.4: if $H(S_i)$<*PT * AVG_LINE_HEIGHT*, S_i is of type 1.
//contains only upper zone
Repeat step 7.
//ignore current strip and go for next strip.

Step 7.5: if $H(S_i)$ > *PM * AVG_LINE_HEIGHT*, S_i will be of type 2, 3, 4, 5, 6, 8, 9 or 10 and will contain at least one headline/meanline and one baseline.

Step 7.6: identify the position of baseline. Mark it as $BASE_{LINE\_NO}$. Also set height of the middle zone as $HGT\_MID=BASE_{LINE\_NO} - H_{LINE\_NO}$.

Step 7.7: set last row of line *LINE_NO* as

$$LR(L_{LINE\_NO})=BASE_{LINE\_NO} + \frac{1}{2}(HGT\_MID).$$

// This will solve the segmentation problem of strip type of category 2, 3, 4, 5, 6, 8, 9.

Step 7.7.1: if (*flag*=1 and *flag1*=1) adjust $FR(L_{LINE\_NO})$=

$$SH_i - \frac{1}{2}(HGT\_MID)$$

//as shown in Fig. 22 we have adjusted the starting //row of 3rd line which was incorrectly segmented by //algorithm *seglines*.

Step 7.8: if $LR(S_i) > LR(L_{LINE\_NO})$
// strip type 10 containing horizontally overlapping //lines
Set $H(S_i)=H(S_i)-(LR(L_{LINE\_NO})-FR(L_{LINE\_NO})$, increment *LINE_NO*, set $FR(L_{LINE\_NO}) = LR(L_{LINE\_NO-1})+1$ and goto step 7.1.
//for same strip.

Step 7.9: if $LR(S_{i+1})<=LR(L_{LINE\_NO})$ increment i
// strip type 7 containing only lower zone
repeat step 7.6.
//for multiple lower zones.

Step 7.10: increment *LINE_NO*, set $FR(L_{LINE\_NO}) = LR(L_{LINE\_NO-1})+1$, go to step 7.
//for next strip.

As shown in Fig. 22, the modified algorithm has segmented line number 2 and 3 in such a way that line number 2 contains some broken part of next line which can be considered as noise for line number 2. We can simply remove this noise on the basis of width of the stroke or location of the noise as it is at the bottom of the line or considering isolated connected components. The problem of line number 3 has been solved, as complete upper zone of this line has been retained in it. Sometimes line number 3 can also have some portion of the lower zone of upper line of big font size. This will be at the top of the line and can be removed on the basis of stroke width or location or considering isolated connected components.

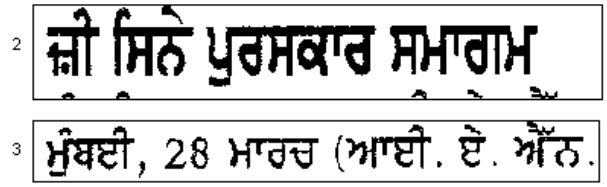

Figure 22. Segmented lines using modified algorithm.

III. Results and Discussions

The algorithm proposed in this paper is useful for segmenting multiple horizontally overlapping lines in printed Indian Scripts. It also joins together the broken components of an over segmented line. Various kinds of strips and the percentage of occurrence of these strips have been calculated for eight scripts as illustrated in the paper. The entire database has been prepared scanning single column news articles from eight printed Indian script newspapers. The algorithm *seglines* segments the overlapping lines accurately 96.45-99.79% times. The overlapping lines in the different sized text in printed newspapers in Gurmukhi script have correctly been

segmented 98.12% of the time. The specific problem in this case has been identified as the situation when there are two consecutive lines of different font size. This problem has successfully been solved in the proposed algorithm.

References

[1] Veena Bansal, *Integrating knowledge sources in Devanagari text recognition*, Ph.D. thesis, IIT Kanpur, INDIA, 1999.

[2] S. Harikumar, K. Jithesh, K. G. Sulochana, R. Ravindra Kumar, "Script based line & character segmentation techniques for Malayalam document images", In *Proceedings of the International Symposium on Machine Translation (iSTRANS 2004)* New Delhi, India, pp. 122-127, 2004.

[3] U. Pal, S. Datta, "Segmentation of Bangla Unconstrained Handwritten Text", In *Proceedings of the ICDAR (ICDAR'03)*, pp. 1128-1132, 2003.

[4] U. Pal, S. Sinha, B. B. Chaudhuri, "Multi-Script Line identification from Indian Documents", In *Proceedings of the ICDAR (ICDAR'03)*, pp. 880-884, 2003.

[5] U. Pal, B. B. Chaudhuri, "Automatic Separation of Machine-Printed and Hand-Written Text Lines", In *Proceedings of the ICDAR (ICDAR'99)*, pp. 645-648, 1999.

[6] J. Dholakia, A. Negi, S. R. Mohan, "Zone Identification in the Printed Gujarati Text", In *Proceedings of the ICDAR (ICDAR'05)*, pp. 272-276, 2005.

[7] B. B. Chaudhuri, U. Pal, "A complete printed Bangla OCR system", *Pattern Recognition*, vol. 31, no. 5, pp. 531-549, 1998,

[8] U. Pal, B. B. Chaudhuri, "Printed Devnagari script OCR system", *Vivek*, vol. 10, pp. 12-24, Jan. 1997.

[9] G. S. Lehal, Chandan Singh, "Text segmentation of machine printed Gurmukhi script", in *Proceedings of SPIE*, vol. 4307, pp. 223-231, 2001.

[10] G. S. Lehal, Chandan Singh, "A technique for segmentation of Gurmukhi text", In *Proceedings of the 9th International Conference on Computer Analysis of Images and Patterns CAIP 2001*, Warsaw, Poland, vol. 2124, pp. 191-200, 2001.

[11] G. S. Lehal, *Optical Character Recognition of Machine Printed Gurmukhi Text***,** Ph.D. thesis, Punjabi University, Patiala, INDIA, 2001.

[12] U. Pal, B. B. Chaudhuri, "Indian Script character recognition: a survey", *Pattern Recognition*, vol. 37, pp. 1887-1899, 2004.

[13] S. Iliescu, R. Shinghal and R. Yee-Mian Teo, "Proposed heuristic procedures to preprocess character patterns using line adjacency graphs", *Pattern Recognition*, vol. 29, no. 6, pp. 951-976, 1996.

[14] B. B. Chaudhuri and U. Pal**, "**Skew angle detection of digitized Indian script documents**",** *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 19, no. 2, pp. 182-186, 1997.

Author Biographies

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 started his carrier as Lecturer in computer application at Jaito centre of Punjabi university, Patiala. At present he is working as Lecturer in Panjab University Regional Centre, Muktsar, Punjab, India. His academic achievements include University Gold medal in Post Graduation. He is currently pursuing Ph.D. degree from Thapar Institute of Engineering and Technology, Patiala, Punjab, India. His research interests include Character Recognition.

Professor Rajendra Kumar Sharma born at Shamil (UP, India) in 1966, received his PhD degree in mathematics from the University of Roorkee (Now, IIT Roorkee), India in 1993. He is currently working as professor at Thapar Institute of Engineering and Technology (TIET), Patiala INDIA, where he teaches, among other things, queuing models and its usage in computer networks. He has been involved in the organization of a number of conferences and other courses at TIET, Patiala. His main research interests are in traffic analysis of Computer Networks, Neural Networks, and Pattern Recognition.

Professor Gurpreet Singh Lehal received undergraduate degree in Mathematics in 1988 from Punjab University, Chandigarh, India, and Post Graduate degree in Computer Science in 1995 from Thapar Institute of Engineering & Technology, Patiala, India and Ph. D. degree in Computer Science from Punjabi University, Patiala, in 2002. He joined Thapar Corporate R&D Centre, Patiala, India, in 1988 and later in 1995 he joined Department of Computer Science at Punjabi University, Patiala. He is actively involved both in teaching and research. His current areas of research are- Natural Language Processing and Optical Character recognition. He has published more than 25 research papers in various international and national journals and refereed conferences. He has been actively involved in technical development of Punjabi and has to his credit the first Gurmukhi OCR, Punjabi word processor with spell checker and various transliteration software. He was the chief coordinator of the project "Resource Centre for Indian Language Technology Solutions- Punjabi", funded by the Ministry of Information Technology as well as the coordinator of the Special Assistance Programme (SAP-DRS) of the University Grants Commission (UGC), India. He was also awarded a research project by the International Development Research Centre (IDRC) Canada for Shahmukhi to Gurmukhi Transliteration Solution for Networking.