International Journal of Image and Graphics
Vol. 9, No. 3 (2009) 321–353

World Scientific
www.worldscientific.com

ON SEGMENTATION OF TOUCHING CHARACTERS AND OVERLAPPING LINES IN DEGRADED PRINTED GURMUKHI SCRIPT

MANISH KUMAR JINDAL
Department of Computer Science and Applications
Panjab University Regional Center
Muktsar, Punjab 152026, India
manishphd@rediffmail.com

GURPREET SINGH LEHAL
Department of Computer Science, Punjabi University
Patiala, Punjab 147002, India
gslehal@gmail.com

RAJENDRA KUMAR SHARMA
School of Mathematics and Computer Applications
Thapar University, Patiala
Punjab 147002, India
rksharma@thapar.edu

Received 7 November 2006
Revised 30 December 2007
Accepted 15 January 2009

Character segmentation plays a very important role in a text recognition system. The simple technique of using inter-character gap for segmentation is useful for fine printed documents, but this technique fails to give satisfactory results if the input text contains touching characters. In this paper, we have proposed two algorithms to segment touching characters, and one algorithm to segment overlapping lines in degraded printed Gurmukhi document. Various categories of touching characters in different zones, along with their solutions, have been proposed. The solution methodology extensively uses the structural properties of Gurmukhi script. The algorithm proposed for segmenting horizontally overlapping lines uses a heuristics based upon the height of a character. The problem of multiple horizontally overlapping lines may occur in a number of situations such as printed newspapers, old magazines and books *etc.* Similarity among Indian scripts allows us to use these algorithms for solving the segmentation problems in other Indian languages also.

Keywords: Gurmukhi script; touching characters; horizontally overlapping lines; top zone; character segmentation.

1. Introduction

Degraded document leads to many problems when processed by an Optical Character Recognition (OCR) system. The main sources of degraded texts are photostatted pages, fax messages, typewriter-printed pages, dot matrix printer printed pages, noisy images, newspaper text, old documents *etc.* Segmentation of characters becomes a complex process in these kinds of documents as it becomes very difficult to decide the touching characters and their position. The position becomes important in order to take a decision to segment them. As such, there is a need for developing algorithms for segmenting the touching characters in degraded printed documents of a language. In the present paper, we have proposed algorithms for segmenting the touching characters in degraded printed documents of Gurmukhi script. It has also been noticed that the problem of overlapping lines is very significant in recognizing the text from newspapers. We have also developed an algorithm for segmenting the horizontally overlapping lines in printed Gurmukhi newspapers. Fine printed leading newspapers of Gurmukhi script also sometimes contain overlapping lines. This is due to the compaction methods used for the printing of the newspapers. Segmentation of these overlapping lines is very important from character recognition point of view. This problem is also significant in the newspapers of other Indian languages. Similarity in some of the Indian scripts allows us to use the algorithm to solve the problem of overlapping lines in other Indian languages.

A number of algorithms[1−6] have been proposed in the past for segmenting touching characters in Roman script. In 1987, Kahan *et al.*[2] have proposed very useful double differential function to segment the touching characters. Tsujimoto and Asada constructed a decision tree for resolving ambiguity in segmenting touching characters.[3] Casey and Nagy have proposed a recursive segmentation algorithm for segmenting touching characters.[4] T. Hong had utilized visual inter-word constraint available in a text image to split word images into pieces for segmenting degraded characters.[5] Liang *et al.*[6] proposed a discrimination function for segmenting touching characters based on both pixel and profile projections. They proposed a dynamic recursive segmentation algorithm with accuracy of 99.40–99.85% for segmenting touching characters. Zhao *et al.*[7] proposed a two stage approach to segment unconstrained handwritten Chinese characters. In their approach, first a character string is coarsely segmented based on the vertical projection and background skeleton, and the blocks of connected characters are identified, then in the fine segmentation stage connected characters are separated with an accuracy of 81.6%.

Many algorithms[8−14] have also been proposed to segment the touching handwritten numerals. Elnagar and Alhajj proposed a thinning based algorithm for segmenting single touching handwritten digits, based on background and contour features in conjunction with a set of heuristics to determine the potential segmentation points with accuracy of 96%.[8] Yu and Yan have developed a method to separate

single touching handwritten numeral strings using structural features.[9,10] In their approach, based on the structural points in the handwritten numeral strings, the touching region of the touching component is determined, and then based on the geometrical information of a special structural point, a candidate touching point is selected. Finally, they used morphological analysis and partial recognition results for the purpose. Chi *et al.*[11] proposed a contour curvature-based algorithm to segment single- and double-touching handwritten digit strings. Lu *et al.*[12] proposed a background thinning approach for the segmentation of connected handwritten digit strings. Pal *et al.*[13] have used water reservoir concept for segmenting unconstrained handwritten connected numerals. They have considered the location, size and touching position (top, middle or bottom) of the reservoir and then analyzed the reservoir boundary, touching position and topological features of the touching pattern, determining the best segmentation point with an accuracy of 94.8%. Chen and Wang used thinning based method to segment single- or multiple-touching handwritten numerals.[14] They performed thinning of both foreground and background regions on the image of connected numeral strings. The end and fork points obtained by thinning are used for cutting points extraction.

Few algorithms[15−21] have been investigated on segmenting the touching characters in Indian scripts. Bansal and Sinha have segmented the conjuncts (one kind of touching patterns) in Devanagari script using the structural properties of the script.[15] Conjuncts are generally found in Devanagari script pages. Conjunct is a combination of a half character followed by a full character. They have segmented the conjuncts with an accuracy of 84%. Their work is, however, limited to the segmentation of conjuncts for touching characters. Garain and Chaudhuri have used a technique based on fuzzy multifactorial analysis to segment the touching characters in Devanagari and Bangla scripts.[16,17] The limitation of their process of segmenting the touching characters is the proposition that at the touching position the width of the touching blob is limited to few columns. This technique works fine if the width of the touching blob is small but it does not work that well when the blob width at touching position is equal to or greater than the stroke width. Although, the simultaneous application of recognition and segmentation process will eliminate this problem to some extent, but performance may not improve drastically. We have tried to resolve this issue in the present paper. Chaudhuri *et al.*[18] have used the principle of water overflow from a reservoir to segment the touching characters in Oriya script. Jindal *et al.*[19] have used the structural properties of Gurmukhi script for segmenting the touching characters in middle zone of printed Gurmukhi script. Lehal and Singh have given an algorithm to segment the touching characters in upper zone of Gurmukhi script.[20,21] Bansal has discussed an algorithm to solve the problem of horizontally overlapping lines in Devanagari script.[22] Harikumar *et al.*[23] have used the concept of average line height to segment the horizontally overlapping lines in Malayalam script. An important survey paper has been published by Srinivas *et al.*[24] on OCR research carried out on Indian scripts.

A special issue of International Journal of Document Analysis and Recognition (IJDAR) has been published on analysis of historical documents containing few very important papers.[25–27] These papers deal with problems erupting in all the major phases like preprocessing, segmentation, feature extraction and classification of an OCR system for recognition of old historical documents. The techniques used in these papers for recognition of historical documents can be explored for recognition of degraded documents of any script. A number of useful methods for line segmentation have been discussed in handwriting segmentation contest[28] held during ICDAR 2007.

In this paper, we have proposed the algorithms for segmenting horizontally overlapping lines and for segmenting touching characters in different zones of Gurmukhi script. The rest of the paper is organized as follows. In the next section, features of Gurmukhi script have been described briefly. In Sec. 3, the problem of horizontally overlapping lines has been discussed and algorithms have been proposed to solve this problem. Various categories of touching characters have been identified in all the three zones of degraded printed Gurmukhi script in Sec. 4. In Sec. 5, we have proposed algorithm for segmenting touching characters in upper zone. Section 6 consists of an algorithm proposed for segmenting touching characters in middle zone. In Sec. 7, we have proposed solution strategy for segmenting touching characters in lower zone. Finally, Sec. 8 contains results and discussions.

2. Features of Gurmukhi Script

Gurmukhi syllabary initially consisted of 32 consonants, three vowel bearers, ten vowel modifiers (including *muktā* having no sign) and three auxiliary signs (semi-vowel). Later on, six more consonants have been added to this script. These six consonants are multi-component characters that can be decomposed into isolated parts. Besides these, some characters modify the consonants once they are appended just below to them. These are called half characters or subjoined characters. The consonants, vowel bearers, additional consonants, vowel modifiers, auxiliary signs and half characters of Gurmukhi script (jointly called sub-symbols in this paper) are given in Fig. 1. In other words, a connected component after removing the headline is called a sub-symbol.

Writing style in Gurmukhi is from left to right. The concept of capital and small characters is not there in Gurmukhi script. A line of Gurmukhi script can be partitioned into three horizontal zones, namely, upper zone, middle zone and lower zone. The middle zone generally consists of the consonants. These three zones are described in Fig. 2 with the help of one example word. The upper and lower zones may contain parts of vowel modifiers, auxiliary signs and half characters. In middle zone, most of the characters contain a horizontal line on the top, as shown in Fig. 1. This line is called the headline. The characters in a word are connected through the headline along with vowel modifiers such as ਿ, ੋ, ਾ *etc.* The headline helps in the recognition of script line positions and character segmentation.

The Consonants

ਸ ਹ ਕ ਖ ਗ ਘ ਙ ਚ ਛ ਜ ਝ ਞ ਟ ਠ ਡ ਢ ਣ ਤ ਥ ਦ ਧ ਨ ਪ ਫ ਬ ਭ ਮ

ਯ ਰ ਲ ਵ ੜ

The Vowel Bearers

ਉ ਅ ਏ

The Additional Consonants (Multi-Component Characters)

ਸ਼ ਖ਼ ਗ਼ ਜ਼ ਫ਼ ਲ਼

The Vowel Modifiers

ੈ ੋ ੇ ੈ ੀ ਿ ਾ ੁ ੂ

Auxiliary Signs (semi-vowels)

ੱ ੰ ਂ

The Half Characters

੍ਹ ੍ਰ ੍ਵ

Fig. 1. Characters and symbols of Gurmukhi script.

Fig. 2. Different zones of Gurmukhi script: (a) upper zone from line number 1 to 2, (b) middle zone from line number 3 to 4, (c) lower zone from line number 4 to 5.

The segmentation problem for Gurmukhi script is entirely different from scripts of other common languages such as English, Chinese, and Urdu *etc.* In Roman script, windows enclosing each character composing a word do not usually share the same pixel values in vertical direction. But in Gurmukhi script, as shown in Fig. 2, two or more characters of same word may share the same pixel values in vertical direction. This adds to the complication of segmentation problem in Gurmukhi script. Because of these differences in the physical structure of Gurmukhi characters from those of Roman, Chinese, Japanese and Arabic scripts, the existing algorithms for character segmentation of these scripts do not work efficiently for Gurmukhi script.

Figures 2(a)–2(c) show the contents of the three zones, i.e. upper, middle and lower zone respectively. The upper zone is either occupied by vowel modifiers,

auxiliary signs or it is empty. Similarly the lower zone is either occupied by vowel modifiers, half characters or it is empty. In Fig. 2, line number two defines the start of headline and line number three, end of the headline. Also, line number four is called the base line.

3. Problem of Horizontally Overlapping Line Segmentation

In printed Gurmukhi script, applying the simple concept of horizontal projection to segment the whole document into individual lines does not work well. Sometimes lower zone characters of one line touch the upper zone characters of next line, thus producing multiple horizontally overlapping lines. Problem of horizontally overlapping lines is very common in Gurmukhi newspapers. Before identifying the problem of multiple horizontally overlapping lines and proposing its solution, we hereby give some definitions:

- **Definition 1** (*Horizontal projection*): For a given binary image of size $L \times M$ where L is the height and M is the width of the image, the horizontal projection is defined as[6]:

$$HP(i), i = 1, 2, 3, \ldots, L$$

  where $HP(i)$ is the total number of black pixels in ith horizontal row.
- **Definition 2** (*Vertical projection*): For a given binary image of size $L \times M$ where L is the height and M is the width of the image, the vertical projection is defined as[6]:

$$VP(j), j = 1, 2, 3, \ldots, M$$

  where $VP(j)$ is the total number of black pixels in jth vertical column.
- **Definition 3** (*Continuous vertical projection*): For a given binary image of size $L \times M$ where L is the height and M is the width of the image, the continuous vertical projection is defined as:

$$CVP(k), k = 1, 2, 3, \ldots, M$$

  where $CVP(k)$ counts the first run of consecutive black pixels in kth vertical column.
- **Definition 4** (*Strip*): A strip can be defined as a collection of consecutive run of horizontal rows, each containing at least one pixel.

The problem of line segmentation further intensifies in degraded printed Gurmukhi script as the horizontal projections of the document divide the whole document into following types of strips:

(i) Strip containing upper zone characters touching with middle zone characters (strip number 1 in Fig. 3).
(ii) Strip containing only upper zone characters (strip number 2 in Fig. 3).
(iii) Strip containing only middle zone characters (strip number 3 in Fig. 3).

Fig. 3. Different strips in printed Gurmukhi text.

(iv) Strip containing only lower zone characters (strip number 4 in Fig. 3).
(v) Strip containing lower zone characters of one line touching with upper zone of next line (strip number 5 in Fig. 3).
(vi) Strip containing two or more horizontally overlapping lines (strip number 6 in Fig. 3).
(vii) Strip containing upper, middle and lower zone characters, i.e. one complete line (strip number 7 in Fig. 3).

These different types of strips make it very difficult to find the category of the given strip. Also, in case of multiple horizontally overlapping lines, it is difficult to estimate the exact position of pixel row that segments one line from the next line. Statistical analysis of newspaper articles reveals the information shown in Table 1, about the occurrence of various strips.

These results have been obtained by analyzing 237 documents, scanned from fine printed newspaper articles. One of these documents is shown in Fig. 3. The percentage of existence of these various kinds of strips has been shown in Table 1. It may be noted that there are seven strips in Fig. 3, but actual number of text

Table 1. Percentage of occurrence of various types of strips in printed newspaper articles.

Type of strip	Percentage of occurrence
1	23.91
2	0.76
3	21.16
4	0.14
5	14.67
6	22.79
7	16.57

lines in this figure is six. Strips 1 and 7 constitute a complete line and need no segmentation. Strips 5 and 6 contain overlapping lines, which require proper segmentation. Strip 2 contains only upper zone characters not touching with middle zone characters and strip 4 contains only lower zone characters not touching with middle zone characters. Both the strips 2 and 4 are generally similar in height and decision has to be made about their identity. Similarly strip number 3, which contains only middle zone, needs its upper and lower zones, if exists. As such, it is necessary to find the exact boundaries of these lines. In the following sub-section, we propose an algorithm to segment the whole document into individual lines.

3.1. *Segmentation of overlapping lines of same size*

Most of the printed area of the newspaper is of same size. We propose the Algorithm 1 for segmenting overlapping lines of same size and associating small strips to their respective text lines of Gurmukhi script. This algorithm segments the whole document into individual lines. The basic idea behind this algorithm is that, in general, the difference between base line of one line and starting line of headline of next line is the same as the height of middle zone and each line has height of lower and upper zone approximately equal to half of the height of middle zone. The input of this algorithm is digitized binary matrix of a single column document of Gurmukhi script and its output is a document with proper line boundaries.

Algorithm 1

Step 1: Find all the n strips in the document using horizontal projections. Mark the first row of strip i as FRS_i and last row of the strip i as LRS_i. Find height of strip i as $HS_i = LRS_i - FRS_i + 1$.

Step 2: Find all the m headlines in the document using horizontal projections. In order to identify the position of headlines, find $MAXPIX = \max\{HP(i)\}$ for $i = 1, 2, 3, \ldots, L$. The headlines are considered as those lines whose $HP(i) \geq 70\%$ of $MAXPIX$
//the threshold limit of 70% is arrived at after detailed and careful
//experimentation.
Whenever $HP(i) > 70\%$ of $MAXPIX$ and $HP(i+1) < 70\%$ of $MAXPIX$, denote it as the ending or last row of the headlines as LRH_1, LRH_2, $LRH_3, \ldots, LRH_m$. Whenever $HP(i) > 70\%$ of $MAXPIX$ and $\mathrm{HP}(i-1) < 70\%$ of $MAXPIX$, denote it as the starting or first row of the headlines as FRH_1, FRH_2, $FRH_3, \ldots, FRH_m$.

Step 3: Find $AVG\_LINE\_HEIGHT = \frac{1}{m-1}\sum_{i=2}^{m}(LRH_i - LRH_{i-1})$.

Step 4: Set $j = 1$ and $FRL_j = FRS_i$. //set first row of first line as first row of first strip.

Step 5: For $i = 1$ to n perform the following operations: //for each strip.
Step 5.1: if $HS_i < 30\%$ of AVG_LINE_HEIGHT, strip is of type 2 then go to step 5. //for next strip.

Step 5.2: if $HS_i > 45\%$ of AVG_LINE_HEIGHT, strip type is 1, 3, 5, 6 or 7 and will contain at least one headline and one baseline.
Step 5.3: identify the location of baseline using continuous vertical projections and mark it as $BASE_j$. Also set height of the middle zone as $HGT\_MID = BASE_j - LRH_j$.
Step 5.4: set last row of line j as $LRL_j = BASE_j + \frac{1}{2}(HGT\_MID)$.
//This will solve the segmentation problem of strip type of category 1, 3, 5, 7.
Step 5.5: if $LRS_i > LRL_j$ then
//strip type 6 containing horizontally overlapping lines
set $HS_i = HS_i - (LRL_j - FRL_j)$ and increment j. Set $FRL_j = LRL_{j-1} + 1$ and go to step 5.1. //for same strip
Step 5.6: while $LRS_{i+1} <= LRL_j$ increment i
//skip current and subsequent strips of type 4 containing only lower zone
Step 5.7: increment j. Set $FRL_j = LRL_{j-1} + 1$ and go to step 5. //for next strip

Step 6: For $j = 1$ to m
display FRL_j to LRL_j as line boundaries.

The proposed algorithm is applied on the document given in Fig. 3. Figure 4 shows the boundaries of different lines identified using this algorithm. This algorithm is developed on the basis that difference between base line of one line and start of headline of next line is same as the height of the middle zone and each line has height of lower and upper zone equal to approximately half of height of the middle zone. This algorithm has shown a remarkable improvement in accuracy, for segmenting the horizontally overlapping lines and associating the small strips to

Fig. 4. Line boundaries identified using proposed algorithm.

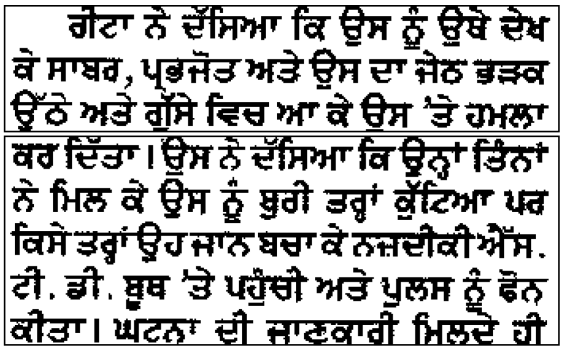


Fig. 5. Document strips containing three and five consecutive horizontally overlapping lines.

their respective lines. This algorithm works efficiently even if the input document contains more than two consecutive horizontally overlapping lines. In the document given in Fig. 5, there are three consecutive horizontally overlapping lines in strip number 1 and five consecutive horizontally overlapping lines in strip number 2. The proposed algorithm segments all the lines of these strips correctly into individual lines.

Table 2 gives a statistics that shows the existence of two and more than two overlapping lines in the scanned newspaper documents. Most of the times only two lines overlap but a total of 27.43% (consisting of overlapping lines) contain more than two overlapping lines. This statistics is again based on the analysis of 237 documents scanned from fine printed newspaper articles.

3.2. *Segmentation of overlapping lines of different sized text*

Documents may contain the text with a large variation in text size. The heading text lines of newspaper document are always larger in size than the actual news text size. One can infer from Fig. 6 that first two lines that are the 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).

Algorithm 1 works accurately for segmenting overlapping lines of the uniform text size. On the contrary, when two different texts sized lines overlap the algorithm does not work that accurately. Figure 7 contains the output of the Algorithm 1 when

Table 2. Percentage of occurrence of two and more than two overlapping lines in newspaper documents.

Number of overlapping lines	Percentage of occurrence
2	72.57
3	12.43
4	10.46
5	4.54

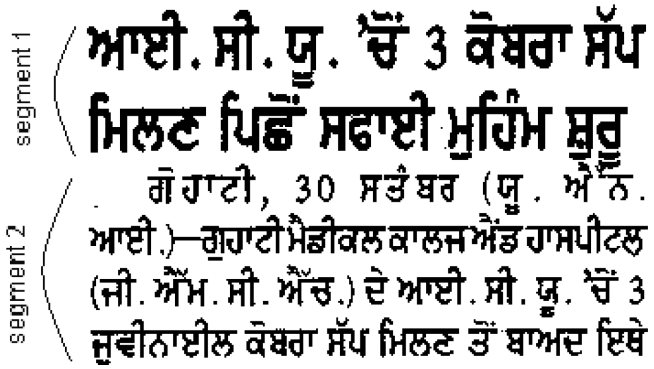
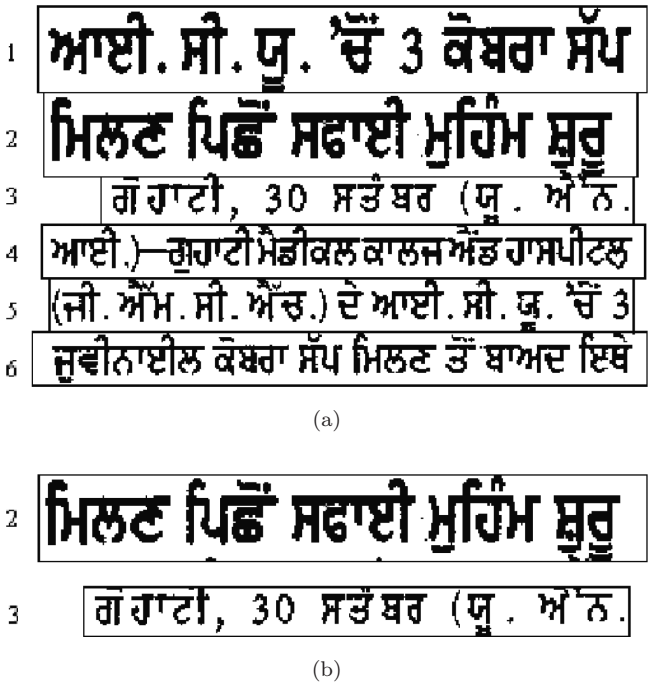


Fig. 6. Printed text of Gurmukhi script containing different size lines.



(a)

(b)

Fig. 7. Line boundaries: (a) six lines identified using Algorithm 1, (b) problem with Algorithm 1 as line number 2 extracts some portion of line number 3.

applied on the document shown in Fig. 6. 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.

We have modified Algorithm 1, in order to solve the problem of segmenting overlapping lines of different size by considering the fact that the number of lines in segment 1 is always 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. We need to modify step 5 to implement this heuristic. The modified step 5 of the Algorithm 1 is given below:

Modification in Algorithm 1

Step 5: For $i = 1$ to m perform the following operations:

Step 5.1: //for strip from segment 1
if $((HS_i > 1.4 * (AVG\_LINE\_HEIGHT))$
set $PT = 35$, $PM = 60$ and go to step 5.4.
Step 5.2: //for strip containing some initial portion from segment 1 followed by some portion from segment 2
if $((HS_i > 1.15 * (AVG\_LINE\_HEIGHT))$
set $PT = 35$, $PM = 60$, *flag* $= 1$ and go to step 5.4.
Step 5.3: //for strip from segment 2
if $((HS_i < (AVG\_LINE\_HEIGHT))$
set $PT = 25$,$PM = 40$ and *flag 1* $= 1$.
Step 5.4: if $HS_i < PT\%$ of *AVG_LINE_HEIGHT*, strip is of type 2 then go to step 5. //for next strip
Step 5.5: if $HS_i > PM\%$ of *AVG_LINE_HEIGHT*, strip type is 1, 3, 5, 6 or 7 and will contain at least one headline and one baseline.
Step 5.6: identify the location of baseline and mark it as $BASE_j$. Also set height of the middle zone as $HGT\_MID = BASE_j - LRH_j$.
Step 5.7: set last row of line j as $LRL_j = BASE_j + \frac{1}{2}(HGT\_MID)$.
//This will solve the segmentation problem of strip type of category 1, 3, 5, 7

Step 5.7.1: if (*flag* $= 1$ and *flag 1* $= 1$)
//current strip is first strip from segment 2
set $FRL_j = FRH_i - \frac{1}{2}(HGT\_MID)$
//as shown in Fig. 8 we have adjusted the starting row of 3rd line
//which was incorrectly segmented by Algorithm 1
set *flag* $= 0$ and *flag 1* $= 0$.

Step 5.8: if $LRS_i > LRL_j$ //strip type 6 of horizontally overlapping lines set $HS_i = HS_i - (LRL_j - FRL_j)$ and increment j. Set $FRL_j = LRL_{j-1} + 1$ and go to step 5.1. //for same strip
Step 5.9: while $LRS_{i+1} <= LRL_j$ increment i
//skip current and subsequent strips type 4 containing only lower zone
Step 5.10: increment j. Set $FRL_j = LRL_{j-1} + 1$ and go to step 5. //for next strip

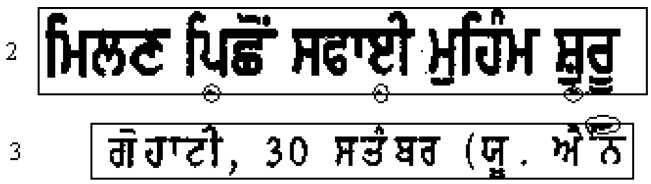


Fig. 8. Different sized lines segmented using modified algorithm.

As shown in Fig. 8, the modified algorithm segments line numbers 2 and line number 3 in such a way that the lost portion of line number 3 has been retained back. Although, line number 2 contains some portion (encircled in Fig. 8) of next line (line number 3) which can be considered as noise for line number 2. But problem here is that line number 3 also contains some portion (encircled in Fig. 8) of line number 2, which can be considered as noise again for line number 3. We can simply remove this noise on the basis of width of the stroke or location of the stroke as it is at the bottom of the line or at the top of the line. The problem of line number 3 has been solved, as complete upper zone of this line has been retained.

3.3. *Segmentation of overlapping lines in other Indian languages*

The problem of overlapping lines has also been found in newspapers of printed Devanagari script. In Devanagari script, we do not find all the strips of seven categories as found in Gurmukhi script (given in Table 1). In this script upper zone and lower zone characters, if present, are always in contact with their middle zone characters. Due to this reason we have found strips in printed Devanagari script newspapers of category/type number 1, 6 and 7 only, i.e. every strip line will contain at least one middle zone. After analyzing 43 documents of Devanagari script from newspapers, we have obtained the following statistics on various types of strips.

It can be seen from Table 3 that problem of horizontally overlapping lines in Devanagari script is very acute as 39.69% of the total strips are of this type. This problem can, however, be solved easily as compared with Gurmukhi script since there are only three kinds of strips in this script. One example document taken from Devanagari newspapers and various strips identified for this document have been shown in Fig. 9.

The Devanagari script does not contain any strip of types 2 and 4 as explained earlier. As shown in Fig. 9, strip number 2 and strip number 3 contain multiple

Table 3. Percentage of occurrence of various strips in Devanagari script printed newspapers.

Type of strip	Percentage of occurrence
1	51.23
5	39.69
6	9.08

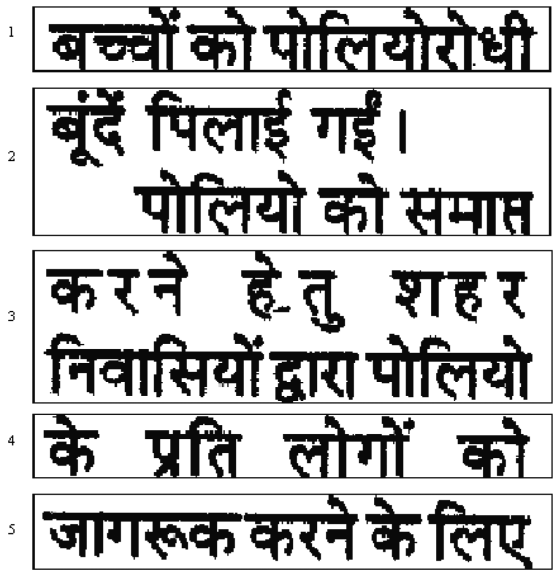


Fig. 9. Text containing overlapping lines from Devanagari newspaper.

horizontally overlapping lines. It has also been analyzed that in Devanagari script whenever there is some vowel modifier in lower zone of one line and next line contains some vowel modifier in upper zone, the two lines almost always overlap. As such, the problem of segmenting horizontally overlapping lines is severe in Devanagari script but due to absence of small strips (type 2 and 4), the problem is comparatively easy to solve in comparison with Gurmukhi script. The algorithm developed for solving the problem in Gurmukhi script works accurately in Devanagari script also and segments the horizontally overlapping lines perfectly.

This is due to the fact that the Algorithm 1 is based on the headline and baseline, which is also present in Devanagari script. Similarly, the modified Algorithm 1 is performing accurately to segment overlapping lines of different size of Devanagari script. Figure 10 contains the output of the Algorithm 1, when applied to the document of Fig. 9.

Bangla is another Indian script that has the concept of headline. The Algorithm 1 proposed in this paper also works accurately in order to segment the horizontally overlapping lines in printed Bangla newspapers. Figure 11 contains an example document from Bangla newspapers.

The problem of horizontally overlapping lines in Bangla script is less acute as compared with Gurmukhi and Devanagari script, as noted from the available newspapers of this script. From the available data, it has been seen that most of the strips in Bangla newspapers are found to be of type 1 and 7 and very few of type 6. The results of proposed algorithm applied on the example document of Fig. 11 are given in Fig. 12.

Fig. 10. Segmented lines of Devanagari newspaper text using Algorithm 1.

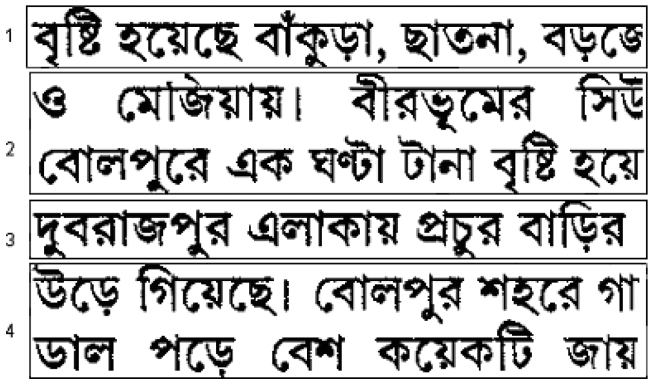


Fig. 11. Bangla text containing overlapping lines.

4. Identification of Various Categories of Touching Characters in Three Zones

Segmentation of words is the next step in segmentation phase. There is generally sufficient amount of space between words, even in degraded documents. Due to this characteristic, segmentation of words is not a complex problem. We have used inter-word gap for word segmentation.

Once the line and word segmentation has been achieved, we have to segment the characters for the purpose of recognizing them. Character segmentation is a

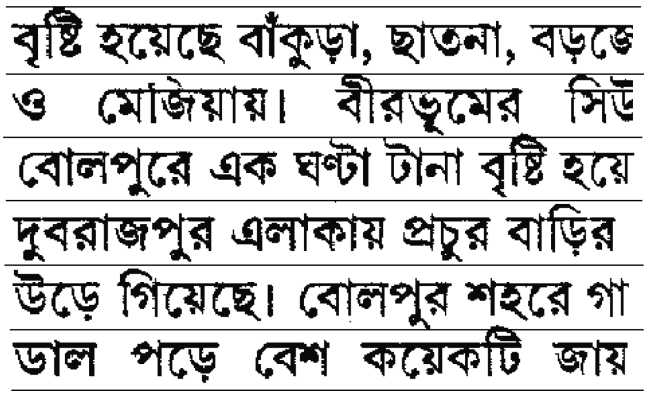


Fig. 12. Bangla text lines segmented using proposed Algorithm 1.

very important step in a text recognition system and accuracy of a text recognition system heavily depends on this step. The importance of character segmentation process increases when the input document is degraded. In this section, we have identified various categories of touching characters in all the three zones in degraded printed Gurmukhi text documents.

In the next sections, we propose algorithms to segment the touching characters of all these proposed categories. The work starts with data collection. For this purpose, we selected truly degraded documents containing touching characters from various books and magazines as well as normal documents, faxed them, copied them and scanned them at 300 dpi resolutions. About 100 such documents were scanned and a database of documents containing touching characters has been created. Figure 13 consists of one paragraph taken from this database. This paragraph contains touching characters in middle, upper and lower zones.

4.1. *Categories of the touching characters in upper zone*

Based on the analysis of Gurmukhi script documents, three categories are proposed for touching characters in upper zone as shown in Table 4. In category a.1, semivowel *bindī* (dot shaped) touches with other characters present in upper zone always on right side. Figure 14(a) contains words from Gurmukhi script in which *bindī*

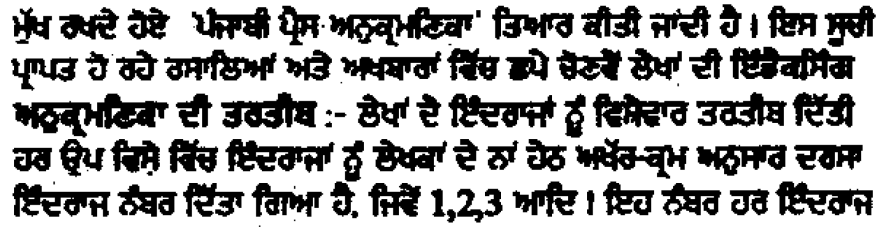


Fig. 13. Gurmukhi text containing touching characters.

Table 4. Various touching categories in upper zone.

Category number	Percentage of touching characters of the category	Structural features of the category	Characters of the category
a.1	35	Dot shaped	*bindī* (ਂ)
a.2	52	Concaved shaped	*adhak* (ੱ)
a.3	13	Convexed shaped	*tippī* (ੰ)

Fig. 14. Gurmukhi words containing touching characters in upper zone (touching characters have been marked with circles): (a) *bindī* touching with other characters, (b) *adhak* touching with other characters, (c) *tippī* touching with other characters.

touches with other characters in upper zone. Analysis of the documents in our database reveals that approximately 35% of the total pairs of touching characters in upper zone fall in this category.

Similarly, for category a.2 concaved shaped *adhak* semi-vowel touches with other characters present in upper zone. Figure 14(b) contains some examples of *adhak* touching with other characters in upper zone. Approximately 52% touching characters out of the total touching characters in upper zone fall in this category.

Also we have identified category a.3 in which *tippī* semi-vowel touches with other characters present in upper zone. Further, it has been revealed from the analysis that the semi-vowel *tippī* always touches with upper zone segment of the two vowel modifiers, i.e. ਿ, ੀ. Figure 14(c) contains examples of *tippī* touching with upper zone segment of ਿ, ੀ in upper zone.

4.2. *Categories of the touching characters in middle zone*

After carefully analyzing the database of touching characters in middle zone, it is found that, on the basis of structural properties of the Gurmukhi script, various touching characters can be classified into five categories. Some characters may fall in multiple categories. For each pair of touching characters, these categories are

Table 5. Various touching categories in middle zone.

Category number	Percentage of touching characters falling in this category	Structural features of the category	Characters	Position numbers in Fig. 15 containing touching pattern of this category
b.1	54	Full sidebar at right end	ਅ ਸ ਖ ਗ ਘ ਜ ਥ ਧ ਪ ਬ ਮ ਯ ਾ	2, 3, 5, 6, 8, 11, 15, 16, 17
b.2	15	Quarter sidebar at right end	ਗ ਰ ਹ ਚ	9, 13, 18
b.3	11	Half sidebar at right end	ਏ ਟ ਦ ਵ ਛ ਢ ਫ ਾ	12
b.4	16	Curved shape at right end	ਉ ਙ ਝ ਠ ਡ ਤ ਨ ਭ ਲ ਵ ੜ	1, 4, 10, 14
b.5	4	Miscellaneous characters	ਕ ਣ	7

Fig. 15. Words containing touching characters in middle zone.

defined on the basis of right side of the left character of the pair. These categories are shown in Table 5.

In category b.1 the characters have full sidebar at its right end. Using statistical analysis, it has been found that 54% of the total pairs of touching characters contain these characters as the left character. There are twelve consonants and one stroke of two vowel modifiers in Gurmukhi script containing full sidebar at right end. Category b.2 contains characters having 75–85% of the full sidebar at their right end. There are four consonants in Gurmukhi script falling in middle zone having this structural feature. It has been observed that approximately 15% characters of the total touching characters fall in this category. The characters in category b.3 contain a half sidebar at right side of the character. Approximate size of the half sidebar is 40–60% of the total height of the character. There are seven consonants and one vowel modifier in Gurmukhi script falling in this category. It has been observed that approximately 11% touching characters of the total touching characters fall in this category. Similarly, we have identified category b.4, in which the touching character contains curved shape at its right end. It has been revealed from the statistical analysis that approximately 16% characters of the total touching characters fall in this category. There are ten consonants in Gurmukhi script falling in this category. Remaining 4% of the touching characters are considered in category b.5. We have two consonants in Gurmukhi script, namely, ਕ and ਣ falling in this category.

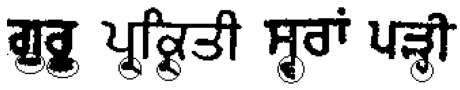
4.3. *Categories of the touching characters in lower zone*

Based on the analysis, we also propose three categories of touching patterns in lower zone as shown in Table 6.

In category c.1, lower zone vowel modifiers and half characters touch the middle zone characters. Depending upon the quality of the input document, approximately 40–70% of the total lower zone vowel modifiers and half characters touch the middle zone characters. This may sometimes happen even with non-degraded texts. Figure 16(a) shows some example words of this kind of touching characters. Category c.2 contains touching pattern in which lower zone vowel modifiers and half characters touches with each other There is a possibility, though rare, of this kind of touching pattern. Figure 16(b) shows one example word containing this kind of touching pattern in lower zone. We have proposed category c.3, in which two components of a multi-component vowel modifier ੋ touch with each other. Figure 16(c)

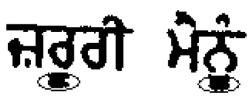
Table 6. Various touching categories in lower zone.

Category number	Percentage of occurrence of touching pattern out of total occurrence of these characters	Characteristics of pattern
c.1	40–70	vowel modifiers and half characters touching with middle zone characters
c.2	0.12	vowel modifiers and half characters touching with each other
c.3	50–70	components of multi-component vowel modifier touching with each other



(a)

(b)



(c)

Fig. 16. Touching characters in lower zone (touching characters have been marked with circles): (a) lower zone characters touching with middle zone characters, (b) lower zone characters touching with each other, (c) two components of multi-component vowel modifier ੋ touching with each other.

shows example words containing this kind of touching pattern. Also it is analyzed that whenever ੁ vowel modifier is present in Gurmukhi document, approximately in 50–70% of the cases the two component of this vowel modifier always touch each other.

5. Segmentation of Touching Characters in Upper Zone

The characters in upper zone usually consist of vowel modifiers, semi-vowels and stroke of vowel bearer. We can divide these vowels and strokes into following three categories:

(i) Vowels present in upper zone only.
(ii) A stroke of vowel present in upper zone and other stroke in middle zone.
(iii) Stroke of a vowel bearer in upper zone.

In the first category, whole of the vowel modifier or semi-vowel lies in upper zone. In second category, there are two vowel modifiers ੀ and ਿ, whose one stroke lies in upper zone and produces touching pattern with the characters of first and third category. The third category contains other shapes appearing in upper zone.

Vowel bearer ੳ contains a stroke which lies in upper zone and may touch with other characters of upper zone. Similarly, when vowel bearer ੳ appears with vowel modifier *hōrā*, it changes the shape of *hōrā* and results in combined shape of ਓ. The *hōrā* vowel modifier present in upper zone of ਓ may touch with other characters in upper zone. Table 7 consists of Pronunciation of name, category and shape of the vowels, semi-vowels and other strokes present in upper zone.

For segmenting the touching characters in upper zone, we have proposed Algorithm 2 based on the structural properties of Gurmukhi characters. Structural properties of Gurmukhi characters reveal that every character in upper zone consists of single concavity or convexity in its structure. This concept of single

Table 7. Pronunciation of name, category and actual shape of vowels, semi-vowels and other strokes (jointly named sub-symbol or character) present in top zone.

Name of the character	Category	Shape of the character
sihārī	second	ਿ
bihārī	second	ੀ
lānvām	first	ੇ
dulānvām	first	ੈ
hōrā	first	ੋ
kanaurā	first	ੌ
adhak	first	ੱ
bindī	first	ਂ
tippī	first	ੰ
ūrā	third	ੁ
hōrā	third	ੂ

concavity or convexity is used to segment the touching characters in upper zone. We propose the following algorithm to segment the touching characters in upper zone.

Algorithm 2

Step 1: Find the top profile of the character and store it in an array *top_profile[]*.
Step 2: Compute first order difference of *top_profile* and store it in Δr_i.
Step 3: Determine the shape of the character (concavity or convexity) as follows:
set $i = 0$;
while ($\Delta r_i = 0$) increment i; //no change in direction skip the column.
if ($\Delta r_i < 0$) shape = concave and go to step 4;
if ($\Delta r_i > 0$) shape = convex and go to step 5.
Step 4: Search for smallest i st $\Delta r_i > 0$ and $\Delta r_{i+1} <= 0$
if no such i exist, then no merged character else mark i as segmentation column.
Go to step 6.
Step 5: Search for smallest i st $\Delta r_i < 0$ and $\Delta r_{i+1} >= 0$
If no such i exist, then no merged character else mark i as segmentation column.
Step 6: If segmentation column exists make a cut at segmentation column.

The working of this algorithm is illustrated using an example word as shown in Fig. 17. One can see from this figure that as the column number increases while moving from left to right, the number of pixels in top profile also increases (Fig. 17(d)). After some columns, the number of pixels starts decreasing. This increase and subsequently decrease of the number of pixels represents the convex shape of the character. Now, whenever this downward trend of the pixels changes its direction to upwards, this marks the segmentation column. If the first character is concave in nature, the number of pixels initially decreases and after few columns it starts increasing. After this increase, whenever a decrement in the number of pixels in a column appears, it is marked as segmentation column.

Segmentation problem in this zone becomes difficult when *kanaurā* touches with other characters. As the shape of this character contains one small concavity followed by small convexity, it produces incorrect segmentation or sometimes even no segmentation. But the chances of occurring touching characters involving *kanaurā* are very less. As shown in Fig. 18, second example word contains *kanaurā* touching with *bindī* and applying Algorithm 2 on this touching pair produces no desired segmentation as shown in Fig. 18(e).

Noise may also sometimes affect the accuracy. During the process of finding the concavity or convexity, if some noise pixels are present in such a way that it disturbs the concavity or convexity of the touching characters, it may also result in incorrect segmentation as happens in third example word of Fig. 18. Another kind of problem is that when two touching characters are merged extremely that

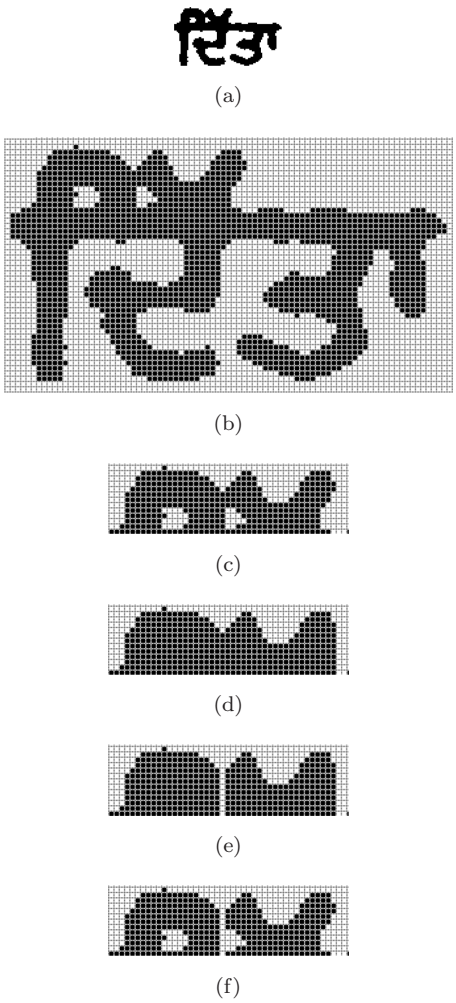


Fig. 17. Segmentation process: (a) example word, (b) extended view of example word, (c) extended view of problem area, (d) top profile of problem area, (e) segmenting column in top profile, (f) actual segmented characters.

individual shapes are hard to extract, in that case also incorrect segmentation takes place as shown in first example word of Fig. 18.

We have tried to implement the same algorithm for segmenting the touching characters in printed Devanagari script. Due to structural similarities between Gurmukhi and Devanagari script, Algorithm 2 with some modification can be implemented for solving the problem.

In Devanagari script, five symbols (vowels/other strokes) in upper zone have convex shape, e.g. ਿ, ੀ, ੇ, ੈ, ੰ and some symbols have convex like shape, e.g. ੱ ੋ ੌ ੰ. Only one vowel modifier ੁ contains two little concavities. Although chances

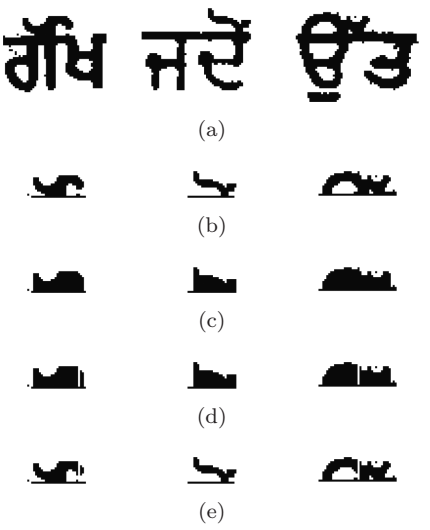


Fig. 18. Segmentation problems in upper zone: (a) example word, (b) problem area, (c) top profile of problem area, (d) incorrect segmentation for first and third words and no segmentation for second word, (e) segmented characters after implementing the Algorithm 2.

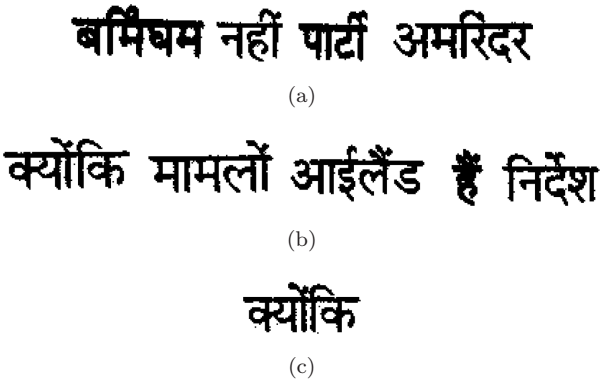


Fig. 19. Touching characters in upper zone of Devanagari script: (a) touching characters which can be segmented using Algorithm 2, (b) touching characters which can be segmented using some modification in Algorithm 2, (c) word containing three touching characters in upper zone.

of ॅ touching with other symbols are very less. The Algorithm 2 works accurately for segmenting the touching characters containing convex shaped symbol as first character (shown in Fig. 19(a)) of the touching pair. Vowel ॆ contains convex like shape and vowel ै contains two small convexities. When Algorithm 2 is applied on touching characters involving ॆ, it will produce no segmentation and for ै will produce incorrect segmentation. Words containing touching characters involving ॆ and ै are shown in Fig. 19(b). So we can modify the Algorithm 2 as whenever there is a significant decrease number of pixels in top profile, and subsequently increase

number of pixels in top profile indicates the segmentation column. Also, when there are three vowels touching in upper zone (shown in Fig. 19(c)), the algorithm will not segment all the three but only first two.

6. Segmentation of Touching Characters in Middle Zone

Most of the touching characters are found in middle zone of a degraded printed Gurmukhi script document. The aforementioned categories in Sec. 4.2 of touching characters in middle zone have been treated individually for segmentation, as detailed below. We propose the following algorithm to segment the touching characters falling in middle zone.

Algorithm 3

Step 1: Recognize all the m headlines as explained in step 2 of Algorithm 1. Denote last row of the headlines as LRH_1, LRH_2, $LRH_3, \ldots, LRH_m$. and first row of the headlines as FRH_1, FRH_2, $FRH_3, \ldots, FRH_m$.
//number of headlines are same as number of lines to identified

Step 2: for $i = 1$ to m perform the following operations:
//m is total number of lines in document
Step 2.1: recognize the boundaries of all p words in line i using vertical projections. Denote first and last column of word j as FCW_j and LCW_j for $j = 1, 2, 3, \ldots, p$
Step 2.2: for $k = 1$ to p performs the following operation:
//for each word k in line i

Step 2.2.1: recognize the headline for individual word using horizontal projections. Denote first row of headline for word k as FRW_k and the last row of the headline for word k as LRW_k.
Step 2.2.2: identify the base line row of the word k using continuous vertical projections. Mark it as $BASE_k$. Also set height of the middle zone as $HGT\_MID = BASE_k - EHW_k + 1$.
Step 2.2.3: note the continuous vertical projection *CVP[w]*, $\forall$ column w from FCW_k to LCW_k, between LRW_k to $BASE_k$.
Step 2.2.4: for $g = FCW_k$ to LCW_k perform the following steps:
//for all the columns in word k

Case 1: //category b.1
Step 2.2.4.1: if number of pixels in *CVP[g]* >= 0.96 $*$ *HGT_MID*
//full sidebar column detected, i.e. category b.1
go to step 2.2.4.2 else go to step 2.2.4.4.
Step 2.2.4.2: while $CVP[g] >= 0.85 * HGT\_MID$, increment g.
Step 2.2.4.3: g marks the segmentation column to segment the touching characters of category b.1. Go to step 2.2.4.
//for next sidebar full, partial or half in the same word if exists

Case 2: //category b.2
Step 2.2.4.4: if number of pixels in *CVP[g]* >= 0.85 * *HGT_MID*
//partial sidebar column detected, i.e. category b.2
go to step 2.2.4.5 else go to step 2.2.4.7.
Step 2.2.4.5: while *CVP[g]* >= 0.75 * *HGT_MID*, increment *g*.
Step 2.2.4.6: *g* marks the segmentation column to segment the touching characters of category b.2. Go to step 2.2.4.

Case 3: //category b.3
Step 2.2.4.7: if number of pixels in *CVP[g]* >= 0.40 * *HGT_MID* and *CVP[g]* <= 0.60 * *HGT_MID*
//half sidebar column detected, i.e. category b.3
go to step 2.2.4.8 else go to step 2.2.4.
Step 2.2.4.8: while *CVP[g + 1])* >= 0.20 * *HGT_MID*, increment *g*.
Step 2.2.4.9: *g* marks the segmentation column to segment the touching characters of category b.3. Go to step 2.2.4.

Step 2.2.5: go to step 2.2. //for next word

Step 3: Go to step 2. //for next line

Case 1 of Algorithm 3 takes care of segmenting the touching characters of a word consisting of touching characters of category b.1. Firstly horizontal and vertical projections of a word having such type of touching characters are taken. After that, start of the headline and end of the headline have been identified in horizontal projection area. The possible locations of sidebar columns are marked in vertical projection area. We can put a white line (segmentation column) after these locations and segmentation is achieved. This algorithm is based upon the structural property of Gurmukhi script, that, in all the Gurmukhi characters if sidebar exists, it is always present at extreme right end of the character, in contrary to Devanagari and Bangla script, where it may appear in the middle of the character. The advantage of this algorithm is that we do not need to identify the candidate for segmentation. Also, more than two touching characters in a single word can be segmented using this algorithm and if the width of touching blob is greater than or equal to the width of the stroke, even then, this algorithm works efficiently.

Similarly case 2 of Algorithm 3 has been developed to segment the touching characters falling in category b.2. The characters falling in this category consist of the sidebar of approximate height of 75–85% of the total height of the character. Whenever such a column occurs, we continue to look for more consecutive columns. When we get a column whose height is less than 75% of the height of the character, we put a segmentation mark for this category of touching characters.

A challenging task in segmenting the touching characters falling in category b.3 is how to identify the small sidebar, which is approximately half of the total height

of the character. We have developed case 3 in Algorithm 3 to segment the touching characters falling in this category. Case 3 of the algorithm sometimes fails producing incorrect segmentation. The reason behind this is that there are some characters in Gurmukhi script, which have little sidebar at their middle or at extreme left end. These characters are ਝ, ਠ, ਨ. A solution for this problem has been implemented by considering the fact that whenever we are encountered in case 3, after terminating of half sidebar columns, it is noted that for the next few columns (depending upon width of the stroke), at least one column must contain less than 20% pixels of height of the characters. If no such column is obtained, ignore that half sidebar column (i.e. it will be from ਝ, ਠ, ਨ characters), otherwise segment the touching characters at this position.

After implementing the above-mentioned cases we look for candidates for segmentation by considering the aspect ratio of the characters. For segmenting the touching characters, we look for the first order difference of the density of pixels from left one-third column to right one-third column of the candidate character. The minimum of the first order difference is considered[2] as segmentation column. Figure 20 shows some words containing touching characters falling in category b.4 with problem area encircled.

After implementing the above-mentioned algorithm, one is able to segment about 76–86% of the total touching characters. Over-segmentation occurs in approximately 8–14% of cases and incorrect segmentation takes place in about 2–3% of cases. Also, in 4–7% cases the algorithm is unable to segment the touching characters and bypasses it without segmenting. The major problems that we face during segmentation using this algorithm are shown in Fig. 21 and explained.

Sometimes a character has a stroke similar in shape of half, full or quarter sidebar, as shown in Fig. 21(a). Since Algorithm 3 (case 1, 2 and 3) is based on the concept of sidebar, it results into over-segmentation, by considering a non-sidebar stroke as sidebar stroke. Identifying the candidate of segmentation is not possible in some cases as shown in Figs. 21(b) and 21(d). This is due to the fact that width of touching characters pair is comparable to the two widest characters in Gurmukhi script (ਘ, ਅ). A solution to this problem has been found, using the fact that both of these characters do not contain any headline. This concept is used to identify whether a wide character is actually a touching pair or a single character (ਘ, ਅ).

The proposed algorithm fails to segment the characters when the touching blob is very big in size as shown in Fig. 21(c). This, however, identifies the touching pair using its aspect ratio.

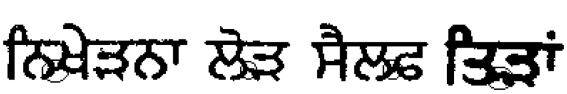
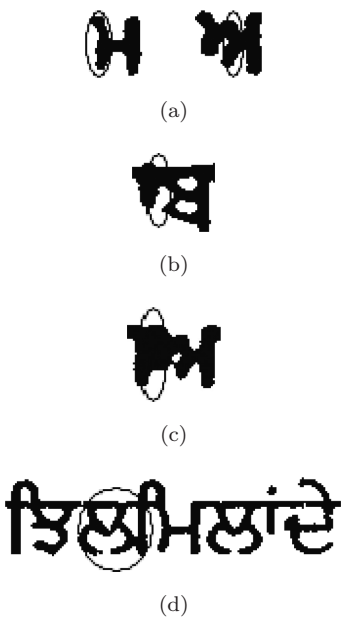


Fig. 20. Touching characters falling in category b.4 (problem area encircled).



(a)

(b)

(c)

(d)

Fig. 21. Problems in segmenting touching characters in middle zone.

7. Segmentation of Touching Characters in Lower Zone

As described in Sec. 4.3, touching characters in lower zone can be divided into three categories. For category c.1 it is sufficient to identify the base line of a strip. Base line segments the middle zone characters from lower zone characters. As shown in Fig. 22(a–d), middle zone characters have been segmented from lower zone characters by identifying the base line. For category c.2, candidate of segmentation can be identified by the aspect ratio of the characters. The character having aspect ration greater than a threshold value is considered to be candidate of segmentation. For actual segmentation, one can use the same technique as used for segmenting the touching characters of category b.4 in Sec. 6. Recognition of pattern of category c.3 is sufficient and needs no segmentation.

8. Results and Discussions

The main objective of this work is to provide a complete solution for segmentation phase including line, word and character segmentation of Gurmukhi script documents. As the problem of word segmentation is trivial and we have discussed in detail character segmentation and line segmentation problems which include methods to segment the multiple horizontally overlapping lines and segment the touching characters present in all the three zones in degraded printed Gurmukhi script.

For segmenting horizontally overlapping lines, we have scanned a number of documents taken from leading newspapers of Gurmukhi script. The problem of

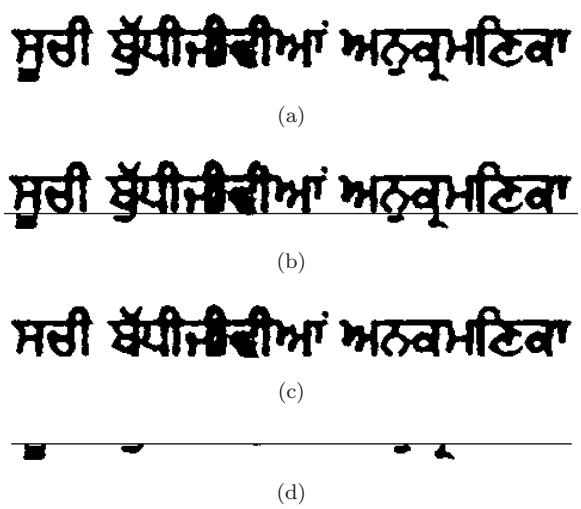


Fig. 22. Segmentation of touching characters of category c.1 in lower zone: (a) line containing touching characters of category c.1, (b) base line identified, (c) middle zone characters separated from lower zone, (d) lower zone characters separated from middle zone characters.

overlapping lines and existence of small strips is found even in good quality newspapers as described in Table 1. The algorithm developed in the present paper segments the horizontally overlapping lines along with associating the small size strips (containing only lower/upper zone) to their respective text lines. A modified version of this algorithm is able to segment the overlapping text lines of different size. The algorithms discussed in literature to segment overlapping lines,[15,16,23] fail to solve the problem of segmenting the overlapping lines of different size. As given in Table 2, the problem of more than two overlapping lines exists in Gurmukhi script documents and the available methods do not solve this problem. The Algorithm 1 proposed by us tackles this problem very efficiently. This algorithm is also tested on documents taken from poorly printed magazines and old documents of Gurmukhi script. We have achieved 98.6% accuracy in segmenting the horizontally overlapping lines along with associating the small size strips (containing only lower/upper zone) to their respective text lines using this algorithm. The same algorithm has also been used to segment the horizontally overlapping lines from Devanagari and Bengali script and it produces equally good results.

We have also tried to solve the challenging problem of segmentation of merged characters in degraded Gurmukhi text. For this purpose, the merged characters are segmented according to the touching zones. Separate algorithms have been developed for the three zones. The algorithms have been tested on around 100 degraded printed Gurmukhi script documents. The results on eight representative degraded printed Gurmukhi script documents are given in Table 8. The percentage accuracy of segmentation for these documents is in the range of 77–86%.

Table 8. Percentage accuracy in middle zone.

Document	Total sub-symbols	Number of touching sub-symbols in category				Number of correctly segmented sub-symbols	Missed segmentation (percentage)	Over-segmentation (percentage)	Percentage accuracy
		b.1	b.2	b.3	b.4				
Doc1	955	39	12	14	2	56	5.97	10.45	83.58
Doc2	1023	46	9	30	13	81	5.10	12.25	82.65
Doc3	984	105	26	21	18	131	7.06	15.88	77.06
Doc4	1023	119	33	41	25	200	3.21	5.05	91.74
Doc5	1045	125	34	36	27	189	4.50	10.36	85.14
Doc6	992	57	7	14	22	86	5.00	9.00	86
Doc7	923	41	6	17	6	58	2.86	14.28	82.86
Doc8	776	14	4	2	15	32	0	8.57	91.43

Table 9. Percentage accuracy in upper zone.

Document	Total characters in upper zone	Number of touching sub-symbols	Number of correctly segmented sub-symbols	Missed segmentation (percentage)	Over-segmentation (percentage)	Percentage accuracy
Doc1	307	19	15	5.26	15.79	78.95
Doc2	357	15	12	0	20	80.0
Doc3	319	22	20	4.55	4.55	90.90
Doc4	297	17	13	5.88	17.65	76.47
Doc5	324	20	16	10	10	80.00
Doc6	267	14	12	0	14.29	85.71
Doc7	243	18	15	5.56	11.11	83.33
Doc8	189	10	8	10	10	80.00

Similarly, for segmenting the touching characters in upper zone we have tested Algorithm 3 proposed in Sec. 7 and observed that an accuracy of 76–86% is achieved. The results are given in Table 9.

For segmenting the touching characters of category c.1 in lower zone, the technique given in Sec. 7 has been tested. It has been observed that 94.4% of lower zone characters are correctly segmented from middle zone characters by using the concept of base line. It has been observed that the touching characters of category c.2 are very few in degraded printed Gurmukhi documents. The proposed algorithm segments these touching characters also very efficiently.

Figure 23 contains part of a Gurmukhi script document containing touching characters. Figure 24 contains results of standard character segmentation algorithms applied on the document. Each sub-symbol has been shown by different color. One can see in Fig. 24 that most of the touching sub-symbols have not been segmented correctly. We have encircled a few non-segmented sub-symbols.

Figure 25 contains the output of the document after the application of Algorithm 1. One can see that most of touching sub-symbols have been correctly

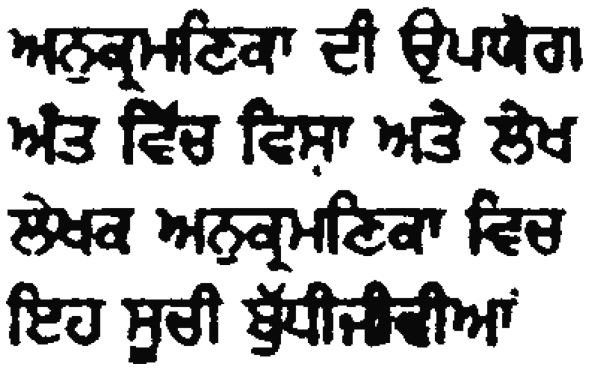


Fig. 23. Part of Gurmukhi script document.

Fig. 24. Segmented characters using standard character segmentation algorithms producing incorrect segmentation.

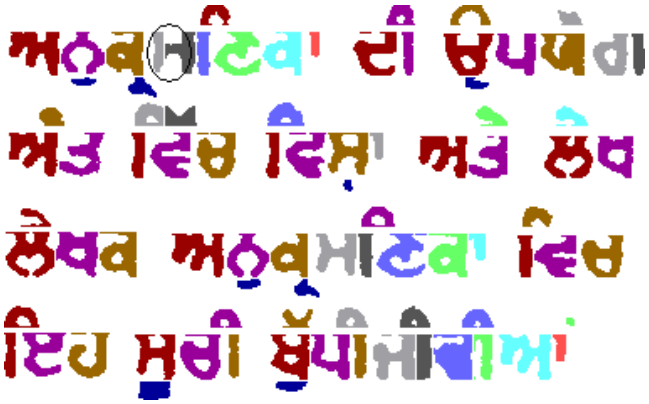


Fig. 25. Segmented characters using proposed algorithms.

segmented. It can also be noted that, one sub-symbol has been over-segmented. The over-segmented character has been encircled.

References

1. Y. Lu, *Patt. Recogn.* **28**(1), 67–80 (1995).
2. S. Kahan, T. Pavlidis and H. S. Baird, *IEEE Trans. on Pattern Analysis and Machine Intelligence* **9**(2), 274–288 (1987).
3. S. Tsujimoto and H. Asada, "Resolving ambiguity in segmenting touching characters," *1st Int. Conf. on Document Analysis and Recognition*, Saint-Malo, France, pp. 701–709 (1991).
4. R. G. Casey and G. Nagy, "Recursive segmentation and classification of composite character patterns," *6th Int. Conf. on Pattern Recognition Proceedings*, Munich, Germany, pp. 1023–1026 (1982).
5. T. Hong, PhD Thesis, Computer Science Department of SUNY at Buffalo (1995).
6. S. Liang, M. Shridhar and M. Ahmadi, *Patt. Recogn.* **27**(6), 825–840 (1994).
7. S. Zhao, Z. Chi, P. Shi and H. Yan, *Patt. Recogn.* **36**(1), 145–156 (2003).
8. A. Elnagar and R. Alhajj, *Patt. Recogn.* **36**(3), 625–634 (2003).
9. D. Yu and H. Yan, *Patt. Recogn.* **31**(12), 1835–1847 (1998).
10. D. Yu and H. Yan, *Patt. Recogn.* **34**(3), 587–599 (2001).
11. Z. Chi, M. Suters and H. Yan, *Optical Engineering* **34**(4), 1159–1165 (1995).
12. Z. Lu, Z. Chi, W. C. Siu and P. Shi, *Patt. Recogn.* **32**(6), 921–933 (1999).
13. U. Pal, A. Belaïd and Ch. Choisy, *Pattern Recognition Letters* **24**(1–3), 261–272 (2003).
14. Y.-K. Chen and J.-F. Wang, *IEEE Trans. on Pattern Analysis and Machine Intelligence* **22**(11), 1304–1317 (2000).
15. Veena Bansal and R. M. K. Sinha, *Patt. Recogn.* **35**(4), 875–893 (2002).
16. U. Garain and B. B. Chaudhuri, *IEEE Trans. on Systems, Man and Cybern. Part C* **32**, 449–459 (2002).
17. U. Garain and B. B. Chaudhuri, "On recognition of touching characters in printed Bangla documents," *Int. Conf. on Document Analysis and Recognition*, Germany, pp. 1011–1016 (1997).
18. B. B. Chaudhuri, U. Pal and M. Mitra, "Automatic recognition of printed oriya Script," *Int. Conf. on Document Analysis and Recognition*, pp. 795–799 (2001).
19. M. K. Jindal, G. S. Lehal and R. K. Sharma, *Transactions on Engineering, Computing and Technology, Enformatika*, **4**, 121–124 (2005).
20. G. S. Lehal and C. Singh, "Text segmentation of machine-printed Gurmukhi Script," *Document Recognition and Retrieval VIII, Proceedings SPIE*, USA, **4307**, 223–231 (2001).
21. G. S. Lehal and C. Singh, "A technique for segmentation of Gurmukhi text," *Computer Analysis of Images and Patterns, Proceedings CAIP 2001*, W. Skarbek (ed.), Lecture Notes in Computer Science, **2124**, Springer-Verlag, Germany, pp. 191–200 (2001).
22. Veena Bansal, PhD Thesis, IIT Kanpur, India (1999).
23. S. Harikumar, K. Jithesh, K. G. Sulochana and R. Ravindra Kumar, "Script based line and character segmentation techniques for Malayalam document images," *Int. Symp. on Mach. Trans. (iSTRANS 2004) Proceedings*, New Delhi, India, pp. 122–127 (2004).
24. B. A. Srinivas, A. Agarwal and C. R. Rao, *Int. Jour. Comp. Sci. and Engg. Sys. (IJCSES)*, **2**(2), 141–153 (April 2008).

25. K. Ntzios, B. Gatos, I. Pratikakis, T. Konidaris and S. J. Perantonis, *IJDAR* **9**, 179–192 (2007).
26. Utpal Grain, Thierry Paquest and Laurent Heutte, *IJDAR* **9**, 47–63 (2007).
27. L. L. Sulem, A. Zahour and B. Taconet, *IJDAR* **9**, 123–138 (2007).
28. B. Gatos, A. Antonacopoulos and N. Stamatopoulos, "ICDAR 2007 Handwriting Segmentation Contest," *9th Int. Conf. on Document Analysis and Recognition*, pp. 1284–1288 (2007).

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 completed his PhD from Thapar University, Patiala, India in 2008. He started his career as a lecturer in computer application at Jaito Center of Punjabi University, Patiala. Currently he is a Reader at Panjab University Regional Center, Muktsar, Punjab, India. His academic achievements include University Gold medal in Post Graduation. His research interests include Character Recognition, and Pattern Recognition.

Gurpreet Singh Lehal received his undergraduate degree in Mathematics in 1988 from Panjab University, Chandigarh, India, and Post Graduate degree in Computer Science in 1995 from Thapar Institute of Engineering & Technology, Patiala, India and his PhD degree in Computer Science from Punjabi University, Patiala, in 2002. He joined the Thapar Corporate R&D Center, Patiala, India in 1988 and later in 1995 he joined the 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 36 research papers in various international and national journals and refereed conferences. He has been actively involved in technical developments 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 Center 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 Center (IDRC) Canada for Shahmukhi to Gurmukhi Transliteration Solution for Networking.

Rajendra Kumar Sharma received his PhD degree in Mathematics from the University of Roorkee (Now, IIT Roorkee), India in 1993. He is currently a professor at Thapar University, 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 Thapar University, Patiala. His main research interests are in traffic analysis of Computer Networks, Neural Networks, and Pattern Recognition.