World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:2, No:9, 2008

Segmentation Problems and Solutions in Printed Degraded Gurmukhi Script

M. K. Jindal, G. S. Lehal, and R. K. Sharma

***Abstract*—**Character segmentation is an important preprocessing step for text recognition. In degraded documents, existence of touching characters decreases recognition rate drastically, for any optical character recognition (OCR) system. In this paper we have proposed a complete solution for segmenting touching characters in all the three zones of printed Gurmukhi script. A study of touching Gurmukhi characters is carried out and these characters have been divided into various categories after a careful analysis. Structural properties of the Gurmukhi characters are used for defining the categories. New algorithms have been proposed to segment the touching characters in middle zone, upper zone and lower zone. These algorithms have shown a reasonable improvement in segmenting the touching characters in degraded printed Gurmukhi script. The algorithms proposed in this paper are applicable only to machine printed text. We have also discussed a new and useful technique to segment the horizontally overlapping lines.

***Keywords*—**Character Segmentation, Middle Zone, Upper Zone, Lower Zone, Touching Characters, Horizontally Overlapping Lines.

I. Introduction

AS a part of the optical character recognition (OCR), character segmentation techniques are applied to word images before individual characters are recognized. The simplest way to segment the characters is to use inter-character gap as segmentation point. However, this technique does not work well if the text to be segmented contains touching characters.

The motivation behind this paper is that in a poor quality text page, degradation leads to many problems such as: adjacent characters can touch one another; a character may be broken into several pieces; random noise or ink smears may make a character distorted. With the presence of such problems, for many word images, it is difficult to correctly determine their identities. Therefore, many recognition errors and uncertainties remain unresolved if the text image is highly degraded. The degraded texts mostly appear in xeroxed pages, fax massages, typewriter-printed pages, dot matrix printed pages, noisy images, images with blur or skew etc. Touching characters is also one kind of degradation that may decrease the recognition results drastically.

M. K. Jindal is with the Panjab University Regional Centre, Muktsar (Punjab) India. (e-mail: mk1_jindal@ yahoo.co.in).

G. S. Lehal is working as Professor, in the Department of Computer Science & Engineering, in Punjabi University, Patiala (Punjab) India. (e-mail: gslehal@lycos.com).

R. K. Sharma is Professor and Head in School of Mathematics and Computer Applications, at Thapar Institute of Engineering & Technology , Patiala(Punjab) India (e-mail: rksharma@tiet.ac.in).

A number of algorithms have been proposed in the past [1-4] for segmenting touching characters in roman script. Kahan *et al.* [2] have proposed very useful double differential function to segment the touching characters. Tsujimoto and Asada [3] constructed a decision tree for resolving ambiguity in segmenting touching characters. Casey and Nagy [4] proposed a recursive segmentation algorithm for segmenting touching characters. T. Hong [5] has utilized visual inter-word constraint available in a text image to split word images into pieces for segmenting degraded English language characters.

Some work has also been done on segmenting the touching characters of Indian languages [6-12]. Veena Bansal and Sinha [6] have segmented the conjuncts (one kind of touching patterns) in Devanagari script using the structural properties of the script. U. Garain and B.B. Chaudhuri [7] have used a technique based on Fuzzy Multifactorial Analysis to segment the touching characters in Devanagari and Bangla scripts. B. B. Chaudhuri, U. Pal and M. Mitra [8] have used the principle of water overflow, from a reservoir, to segment the touching characters in Oriya script. M.K. Jindal *et al.* [10] have used the structural properties for segmenting the touching characters in middle zone of printed Gurmukhi script. Lehal and Singh [11-12] have also tried to segment the touching characters in upper zone of Gurumukhi script.

In this paper, we have proposed new strategies to segment touching Gurmukhi script characters. First we have developed an algorithm to segment the multiple horizontal overlapping lines in printed Gurumukhi script. These horizontally overlapping lines are found even in clean printed books, magazines and newspapers. At the outset, a database has been prepared after scanning a number of poor quality printed documents containing 20-30% touching characters. Then all the touching locations were carefully analyzed and various categories are proposed based on the structural properties of the Gurmukhi characters. After that, algorithms have been developed to segment the touching characters in middle, upper and lower zone.

II. Characteristics of Gurmukhi Script

Gurmukhi script alphabet consists of 41 consonants and 12 vowels as shown in Fig. 1. Besides these, some characters in the form of half characters are present in the feet of characters. Writing style is from left to right. The concept of upper/lower-case characters is absent in Gurmukhi. 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 zones are shown

in Fig. 2. The upper and lower zones may contain parts of vowel modifiers and diacritical markers.

In Gurmukhi Script, most of the characters, as shown in Fig.1, contain a horizontal line at the upper of the middle zone. This line is called the headline. The characters in a word are connected through the headline along with some symbols as i, I, A etc. The headline helps in the recognition of script line positions and character segmentation. 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 share the same pixel values in horizontal direction. But in Gurmukhi script, as shown in Fig. 2, two or more characters/symbols of same word may share the same pixel values in horizontal 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 does not work efficiently for Gurmukhi script.

Consonants				
u	a	e	s	h
c	k	g	G	L
C	x	j	J	M
t	T	D	Q	N
V	W	d	Y	n
p	f	b	B	m
y	r	l	v	R
S	z	K	F	Z
Pl				
Vowels in Upper zone				
*	E	&	>	~
O	:			
Vowels in Upper and Middle zone				
i	I			
Vowels in Middle zone				
A				
Vowels in Lower zone				
U	<			
Half characters in Lower zone				
H	q	X		

Fig. 1 Gurmukhi script characters and symbols

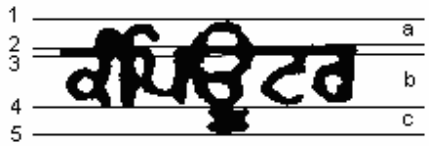

Fig. 2 a) Upper zone from line number 1 to 2, b) Middle Zone from line number 3 to 4, c) lower zone from line number 4 to 5

Fig. 2(a), 2(b) and 2(c) show the contents of the three zones, i.e., upper, middle and lower zone respectively. The upper and lower zones can be empty for a word, but only the vowels/half characters may be present in these zones. In Fig. 2, line number 2 defines the start of headline and line number 3 end of the headline. Also, line number 4 is called the base line.

III. Preprocessing

Preprocessing is applied on the input binary document in order to minimize the effect of spurious noise in the subsequent processing stages. In the present study, both salt and peeper 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 algorithms proposed in the present study do not perform very well in case the image is skewed.

IV. Line Segmentation

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 by [6] as:

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

where $HP(i)$ is the total number of black pixels in i^{th} 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:

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

where $VP(j)$ is the total number of black pixels in j^{th} 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 has been defined as:

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

where $CVP(k)$ counts the first run of consecutive black pixels in k^{th} 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.

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 touches the upper zone characters of next line, thus producing multiple horizontally overlapping lines. This problem further intensifies in printed Gurmukhi script as the horizontal projections of the document, divides the whole document into following categories of strips:

1. Two or more horizontally overlapping lines (strip number 1 in Fig. 3)
2. Only lower zone characters. (strip number 2 in Fig. 3)
3. Only upper zone characters(strip number 3 in Fig. 3)
4. Only middle zone characters(strip number 4 in Fig. 3)
5. Upper, middle and lower zone characters, i.e., complete one line (strip number 5 in Fig. 3)

6. Upper zone characters with middle zone characters(strip number 6 and 8 in Fig. 3)
7. Lower zone characters touching with upper zone of next line (strip number 7 in Fig. 3)

These different kinds 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, which segments one line from the next line. Statistical analysis of newspaper articles reveals the following information.

TABLE I
PERCENTAGE OF OCCURRENCE OF VARIOUS STRIPS

Type of strip	% of occurrence
1	17.54
2	21.49
3	0.88
4	1.31
5	12.28
6	31.57
7	14.91

These results have been obtained by analyzing 54 documents, scanned from fine printed newspaper articles. One of the documents is shown in Fig. 3.

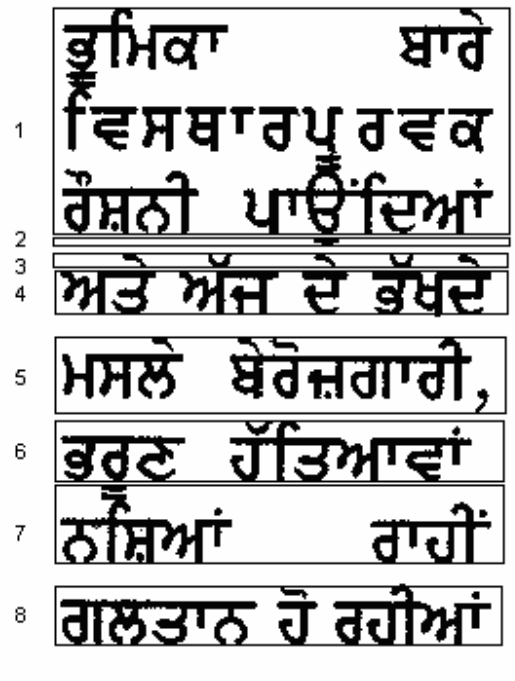

Fig. 3 Strip lines in printed Gurmukhi text

Fig. 3 contains eight strips. It can be seen that actual number of lines in Fig. 3 is also eight (a line contains its upper, middle and lower zone). Except strip number five and eight, no other strip represents a complete line. As such, it is necessary to find the exact boundaries of all the lines. An algorithm, as given below, has been developed to segment this kind of document into individual lines.

Algorithm 1

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, …, S_m. Also denote first row of strip as $FR(S_p)$, last row of strip as $LR(S_p)$and 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 a document are shown in Fig. 3.

Step 2: In order to identify the location of headlines, find

$MAXPIX = \max \{HP(i)\}$, $i = 1, 2, 3, \ldots, L$

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). Let us denote the ending location of the headlines as H_1, H_2, H_3, …, H_n. Also denote the lines to be identified as L_1, L_2, $L3$, …, L_n (number of headlines is same as number of lines)

Step 3: Define

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

Step 4: 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 5: For $i=1$ to m perform the following operations:

{

Step 5.1 : if $H(S_i)$ < 30% of AVG_LINE_HEIGHT, S_i is of type 3(contains only upper zone), repeat step 5(ignore current strip and go for next strip).

Step 5.2: if $H(S_i)$ > 50% of AVG_LINE_HEIGHT, S_i will be of type 1 or 4 or 5 or 6 or 7 and will contain at least one headline and one baseline.

Step 5.3: identify the location of baseline by noting the $CVP(k)$, $\{k=H_{LINE\_NO}$ to $LR(S_i)\}$. The location where $CVP(k)$ ends, mark it as α, every time. The row in which maximum α are found is considered to be the baseline. Mark it as $BASE_{LINE\_NO}$. Also set height of the middle zone as $HGT\_MID = BASE_{LINE\_NO} - H_{LINE\_NO}$.

Step 5.4 : set last row of line $LINE\_NO$ as $LR(L_{LINE\_NO}) = BASE_{LINE\_NO} + \frac{1}{2}(HGT\_MID)$.(case number 4,5,6,7 solved here)

Step 5.5: if $LR(S_i) > LR(L_{LINE\_NO})$, (case 1 of horizontally overlapping lines). Set $H(S_i)=H(S_i)-(LR(L_{LINE\_NO})-FR(L_{LINE\_NO})$, $LINE\_NO = LINE\_NO + 1$. Also Set $FR(L_{LINE\_NO})= LR(L_{LINE\_NO-1})+1$ and goto step 5.1(for same strip)

Step 5.6: if $LR(S_{i+1})<=LR(L_{LINE\_NO})$ set $i=i+1$ (case 2 of only lower zone). Repeat step 5.6(for multiple lower zones)

Step5.7: $LINE\_NO = LINE\_NO+1$. $FR(L_{LINE\_NO})= LR(L_{LINE\_NO-1})+1$ Go to step 5(for next strip)

}

Step 6: for j=1 to $LINE\_NO$

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

END.

Fig. 4 Different line boundaries identified using proposed algorithm

Fig. 4 shows the boundaries of different lines identified using the proposed algorithm. This algorithm has shown a remarkable improvement in accuracy, for segmenting the horizontally overlapping lines and associating the small strips to their respective lines. This algorithm works even if the input document contains many horizontally overlapping lines. As shown in Fig. 5, there are six consecutive horizontally overlapping lines in strip number one. The proposed method segments all the lines of this strip correctly into individual lines. We have obtained 95% accurate results of the algorithm for segmenting the multiple horizontally overlapping lines and associating the small strips to their respective lines using this algorithm.

V. Identification of Touching Characters

In this section, we propose an efficient algorithm to segment the touching characters in a degraded text document. The work for the proposition of this algorithm starts with data collection.

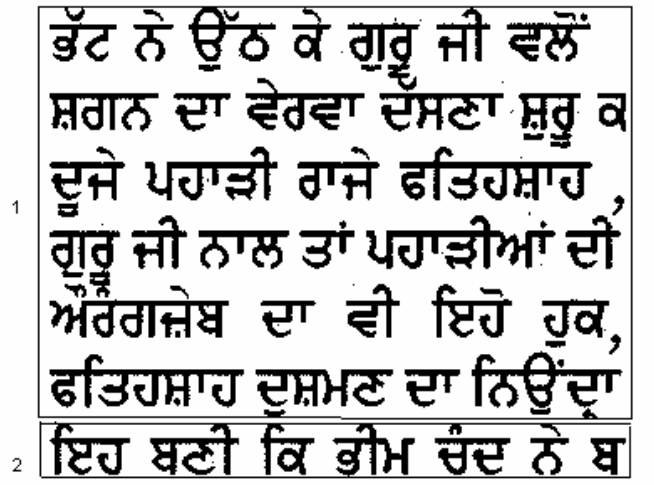

Fig. 5 Document strip containing six consecutive horizontally overlapping lines

A. Data Collection

Data collection is a time consuming task. We selected true 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 which contains almost 6000 touching characters, thus a sufficiently large database of touching characters has been created. Fig. 6 shows one paragraph taken from this database. This paragraph contains touching characters in middle, upper and lower zone.

B. 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

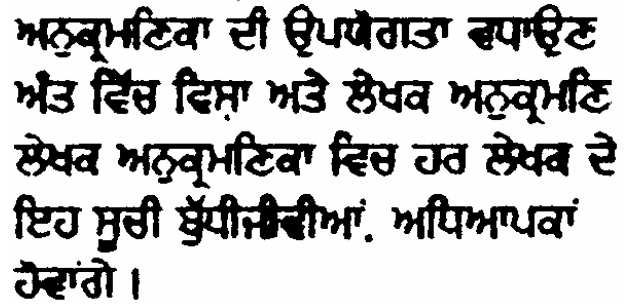

Fig. 6 Gurmukhi paragraph containing touching characters

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 defined on the basis of left character of the pair. These categories are hereby briefly described.

Category 1: Touching characters containing full sidebar at right end

Using statistical analysis, it is found that 54% of the total pairs of touching characters contain these characters at left side, which have full sidebar at their right end. There are total

twelve consonants and one stroke of two vowels in Gurmukhi script containing sidebars at right end, as mentioned below:
a, s, k, g, G, j, W, Y, p, b, m, y, .

For example, in Fig. 7 touching characters at positions marked as 1, 3, 5, 6, 7, 9, 10, 12, 13, 14, 17, 18 and 19 are from this category.

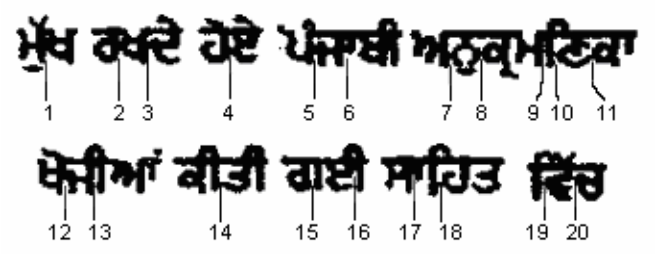

Fig. 7 Words containing touching characters in middle zone

Category 2: Touching characters containing partial sidebar at right end

There are four consonants in Gurmukhi script falling in middle zone, that do not have full sidebar at their extreme right end, but it contains 75-85% of the full sidebar. It has been observed that approximately 15% characters of the total touching characters fall in this category. These characters are: g, r, h, C . In Fig. 7 touching characters at positions marked as 2, 4 and 15 belong to this category.

Category 3: Touching Characters containing little sidebar at right end

It has been observed that approximately 11% characters of the total touching characters fall in this category. In this category, the characters contain a little sidebar at right side of the character. Approximate size of the sidebar is half of the total length of the character. There are seven consonants and one vowel in Gurmukhi script falling in this category and these are: e, t, d, v, x, M, f, A . Fig. 7 contains touching characters at 16th and 20th position, from this category.

Category 4: Touching Characters containing curved shape at right end

It has been revealed from the analysis that approximately 16% characters of the total touching characters fall in this category. Here, the touching character contains curved shape at right extreme end. There are ten consonants in Gurmukhi script, namely: L, J, T, D, V, l, B, R, u, n falling in this category. Fig. 7 contains touching character at position 8 from this category.

Category 5: Touching Characters of miscellaneous type

Rest, 4% of the total touching characters fall in neither of the above mentioned categories. So these are categorized in this category containing touching characters of miscellaneous type. There are two consonants in Gurmukhi script, namely: c, N falling in this category. Fig. 7 contains touching character at position 11 from this category.

C. Categories of the touching characters in upper zone

Following three categories are being proposed in the upper zone for touching characters.

Fig. 8 Gurmukhi words containing touching characters in upper zone (touching characters have been marked with circles). a) *Bindi* touching with other characters, b) *Adhak* touching with other characters, c) *Tippi* touching with other characters

Category 1: Bindi (:) touching with other characters

By carefully analyzing, it is found that 35% of the total pair of touching characters in upper zone fall in this category. In this category, vowel "*Bindi*" (dot shaped) touches with other characters present in upper zone either form left or right side. Fig. 8(a) contains words from Gurmukhi script in which *Bindi* touches with other characters in upper zone.

Category 2: Adhak (&) touching with other characters

Approximately 52% touching characters of the total touching characters in upper zone fall in this category. In this category, "*Adhak"* vowel touches with other characters present in upper zone. Fig. 8(b) contains some examples of *Adhak* touching with other characters in upper zone.

*Category 3: Tippi (*) touching with other characters*

It has been seen that 13% touching characters of the total touching characters in upper zone fall in this category. In this category, *Tippi* vowel touches with other characters present in upper zone. Further, it has been revealed from the analysis that the vowel *Tippi* always touches with upper zone segment of the vowels i, I. Fig. 8(c) contains examples of *Tippi* touching with upper zone segment of i, I in upper zone.

D. Categories of the touching characters in lower zone

Based on the analysis, we consider the following two categories in lower zone.

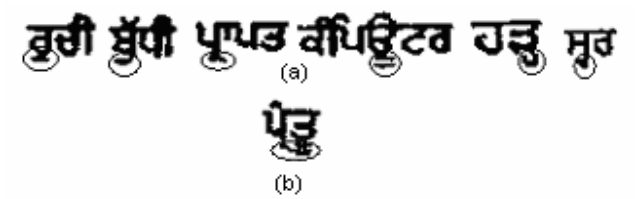

Fig. 9 touching characters in lower zone (touching characters have been marked with circles), a) Lower zone characters touching with upper zone, b) lower zone characters touching with each other

Category 1: Lower zone vowels and half characters touching with middle zone characters

Depending upon the quality of the input document, approximately 40-70 % of the total lower zone vowels and half characters always touch the middle zone characters. This may sometimes happen even with non-degraded texts. Fig. 9(a) shows some examples of this kind of touching characters.

Category 2: Lower zone vowels and half characters touching with each other

There is a possibility though rare of lower zone vowels touching with each other. Fig. 9(b) shows this kind of touching pattern in lower zone.

VI. Segmentation in Middle Zone

Most of the touching characters are found in middle zone of a degraded Gurumukhi script document. The afore mentioned categories of touching characters in middle zone are treated individually for segmentation, as detailed below. We have devised following algorithms to segment the touching characters falling in middle zone.

Algorithm 2
BEGIN
Step 1: Recognize the headline. In order to identify the location of headlines, find
$MAXPIX$= max {$HP(i)$}, i = 1, 2, 3, …, 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). Let us denote starting location of headlines as SHL_1, SHL_2, SHL_3, …, SHL_n and the ending location of the headlines as EHL_1, EHL_2, EHL_3, …, EHL_n.
Step 2: for i= 1 to $LINE\_NO$ (where $LINE\_NO$ denotes the total number of lines in the input binary document as found in algorithm 1) repeat the following steps:
{
Step 2.1: Recognize individual words by considering $VP(j)$ for j = 1, 2, 3, …, M, from $FR(L_i)$ to $LR(L_i)$(first row and last row of i^{th} line denoted as $FR(L_i)$ and $LR(L_i)$. Whenever $VP(j)$=0, it denotes a word boundary. Denote the individual words as W_1, W_2, W_3, …, W_p. First and last column of each word are denoted as $FC(W_j)$ and $LC(W_j)$, j = 1, 2, 3,…, p.
Step 2.2: for k=1 to p performs the following operation:
{
Step 2.2.1: Recognize the headline for individual word. For that find $HP(t)$, {t=$SHL(L_i)$-4 to $EHL(L_i)$+4 }, between $FC(W_k)$ to $LC(W_k)$. find
$MAXPIX1$= max {$HP(t)$} , t=$SHL(L_i)$-4 to $EHL(L_i)$+4
The headlines are considered as those lines whose $HP(t) \geq$ 90% of $MAXPIX1$. Let us denote starting location of headline for word k as FHW_k and the ending location of the headline for word k as LHW_k.
Step 2.2.2 : identify the base line row of the word k by noting the $CVP(j)$, {j=FHW_k to $LR(L_i)$}. The location where CVP (j) ends, mark it as α, every time. The row in which maximum α's are found is considered to be the baseline. Mark it as $BASE_k$. Also set height of the middle zone as HGT_MID = $BASE_k$ – LHW_k.
Step 2.2.3: Note the Continuous vertical projection $CVP(m)$, m=$FC(W_k)$ to $LC(W_k)$, between EHW_k to $BASE_k$
Step 2.2.4: for g = $FC(W_k)$ to $LC(W_k)$perform the following steps:
{
Case 1 :(category 1)
Step 2.2.4.1: if number of pixels in $CVP(g)$ >= (96/100) *HGT_MID(Full sidebar column detected, first category) go to step 2.2.4.2 else go to step 2.2.4.4.
Step 2.2.4.2: while $CVP(g)$>= 85*HGT_MID/100,g=g+1
Step 2.2.4.3:g marks the column where segmentation point to be inserted to segment the touching characters of first category. Go to step 2.2.4 //for next sidebar (full, quarter or half) in same word.
Case 2 : (Category 2)
Step 2.2.4.4: if number of pixels in $CVP(g)$ >= (85/100) *HGT_MID(quarter sidebar column detected, Fourth category) go to step 2.2.4.5 else go to step 2.2.4.7
Step 2.2.4.5: while $CVP(g)$>= 75*HGT_MID/100, g = g+1
Step 2.2.4.6 : g marks the column where segmentation point to be inserted to segment the touching characters of second category. Go to step 2.2.4 //for next sidebar in same word.
Case 3 : (Category 3)
Step 2.2.4.7: if number of pixels in $CVP(g)$ >= (40/100) *HGT_MID and $CVP(g)$ <= (60/100) *HGT_MID (half sidebar column detected, third category) go to step 2.2.4.8 else go to step 2.2.4
Step 2.2.4.8: while CVP (g+1) >= 20*HGT_MID/100, g=g+1
Step 2.2.4.9:g marks the column where segmentation point to be inserted to segment the touching characters of third category. Go to step 2.2.4 //for next sidebar in same word.
}
Step 2.2.5: go to step 2.2 //for next word
}
Step 3: go to step 2. // for next line
END

A. Solution for segmenting touching characters falling in first category

For segmenting the characters of a word having touching characters of the first category, we have developed case 1 of algorithm 2.

Fig. 10 Horizontal & Vertical Projection of a touching word

Fig. 11 White dots showing start of headline, end of headline and possible locations of sidebar Columns

Horizontal and vertical projections of a word having touching characters are given in Fig. 10. Also, start of the headline and end of the headline in Fig. 11 have been marked by white marks in horizontal projection area. The possible locations of sidebar columns in Fig. 11 are marked by white marks in vertical projection area. We can put a white line after these locations and segmentation is achieved as shown in Fig. 12.

Fig. 12 Touching characters segmented using case 1 of algorithm 2

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 with Devanagari and Bangla script, where it may be 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.

B. Solution for segmenting touching characters falling in Second category

Case 2 of algorithm 2 has been developed to segment the touching characters falling in this category. The characters falling in second category consists of the sidebar of height more than 85% of the total height of the character. Whenever such a column occurs, we continue for looking 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.

C. Solution for segmenting touching characters falling in third category

A challenging task in segmenting the touching characters falling in this category is how to identify the little sidebar, which is approximately half of the total height of the character. We have developed case 3 in algorithm 2 to segment the touching characters falling in this category.

Case 3 of the algorithm sometimes fails producing over 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 L, T, n . 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 next 3-4 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 found, ignore that half sidebar column (it will be from L, T, n characters) otherwise segment the touching characters at this position.

D. Solution for segmenting characters falling in fourth category

After implementing the above mentioned algorithm we look for candidate of segmentation by considering the aspect ratio of the characters. Now for segmenting the touching characters of these candidates, we look for the density of the pixels in columns from left one third to right one third columns of the candidate character. Wherever the density of pixels is minimum we consider it as segmentation column. Fig. 13 shows some words containing touching characters falling in fourth category and the problem areas have been encircled.

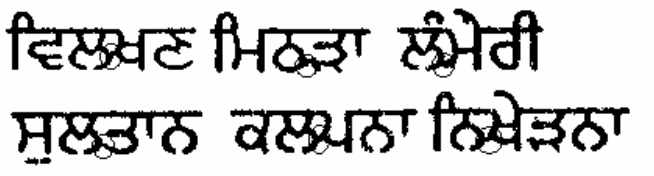

Fig. 13 Touching characters falling in fourth category (problem area encircled)

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 algorithms is unable to segment the touching pair and bypasses it without segmenting. The major drawbacks and problems, we face during segmentation using this algorithm are shown in Fig.14 and explained as below:

Sometimes a character has a stroke similar in shape of half, full or quarter sidebar, as shown in Fig. 14(a). Since algorithm 2 (case 1, 2 and 3) is based on the concept of sidebar, it results 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 Fig. 14(b) and 14(d). This is due to the fact that width of touching characters pair is

comparable to the two widest characters in Gurumukhi script (G, a). A solution to this problem has been found as both of these characters do not contain any headline. So this concept can be used to identify that weather a wide character is actually a touching pair or a single character (G, a).

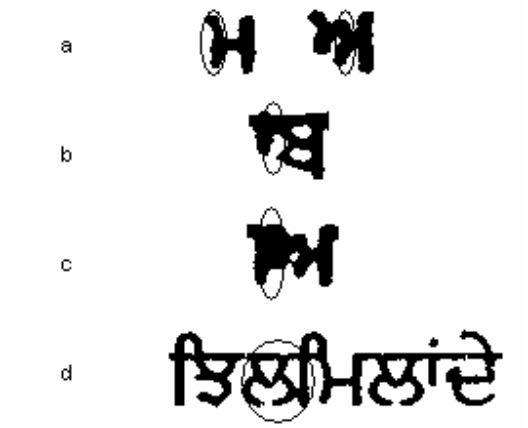

Fig. 14 Problems in segmenting characters in middle zone

Similarly, as shown in Fig. 14 (c) even though, it is identified that this character is a touching pair using its aspect ratio, but touching blob is very much big and it results in wrong segmentation.

VII. Segmentation In Upper Zone

We can divide the vowels into following four categories.

1. Vowel present in upper zone only.
2. Vowel present in upper and middle zone.
3. Vowel in middle zone only.
4. Vowel present in lower zone only.

TABLE II
NO OF VOWELS FALLING IN EACH CATEGORY

Categories of vowel	Number of vowels
First	7
Second	2
Third	1
Fourth	2

The pronunciation, actual shape and examples of the vowels, falling in first category are shown in Fig. 15 and that of falling in second category are shown in Fig. 16.

Except the above mentioned vowels falling in upper zone there are some characters whose one stroke falls in upper zone. The character pronunciation, the stroke of the character falling in upper zone and example words are shown in Fig. 17.

Fig. 15 Pronunciation, actual shape and example words of the vowels falling in first category

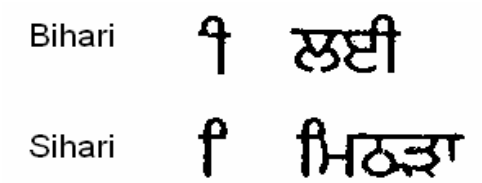

Fig. 16 Pronunciation, actual shape and example word of the vowels falling in second category

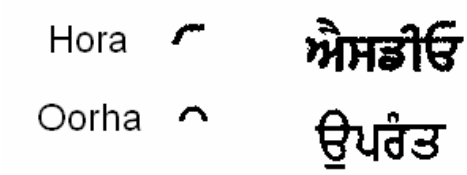

Fig. 17 Pronunciation, actual shape and example words of some character strokes in upper zone

For segmenting the touching characters in upper zone, we have developed a strategy based on the structural properties of Gurumukhi characters in upper zone. 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 concavity or convexity is used to segment the touching characters in upper zone.

Algorithm 3
BEGIN
Step 1: Using the vertical projection in upper zone identify the boundaries of each character. For that whenever $VP(i)=0$ for $i = 1, 2, 3, \ldots, L$, it is marked as the boundary of character. Let us denote the different

characters as C_1, C_2, …, C_n. Denote first column of the character as FC_1, FC_2, …, FC_n and Last column of the character LC_1, LC_2, …, LC_n.

Step 2: for k=1 to n performs the following operations (for each character in upper zone)

Step 2.1: find the top profile of the character. For that, for $j=FC_k$ to LC_k perform the following

Step 2.1.1: mark the row as X, in which first black pixel in j^{th} column is encountered. Now calculate $TP(j)=LR-X+1$ where TP is top profile and LR is last row of upper zone.

Step 2.2: for $j = FC_k$ to LC_k perform the following

Step 2.2.1: if $TP(j+1)>=TP(j)$ go to step 2.2.3(concavity) else goto step 2.2.5(convexity)

Step 2.2.2 : while $TP(j+1)>= TP(j)$ & $j<LC_k$ increment j and repeat step 2.2.2

Step 2.2.3: while $TP(j+1)<=TP(j)$ & $j<LC_k$ increment j and repeat step 2.2.3.

Step 2.2.4: if $j <= LC_k$, j marks the segmentation column

Step 2.2.5 : while $TP(j+1)<=TP(j)$ & $j<LC_k$ increment j and repeat step 2.2.5

Step 2.2.6: while $TP(j+1)>=TP(j)$ & $j<LC_k$ increment j and repeat step 2.2.6.

Step 2.2.7: if $j <= LC_k$, j marks the segmentation column

END.

In this algorithm, initially, top profile of each character in upper zone is identified shown in Fig. 18(c). The top profile is scanned from left to right to examine the presence of concavity/convexity. For example, in first word of Fig. 18, one can see that as the column number increases while moving from left to right the number of pixels in top profile also increases. After few columns, number of pixels starts decreasing. This increase and subsequent decrease of the number of pixels represents the convex shape of the character. Now, going further, whenever the downward trend of the number of pixels changes its direction to upwards that marks the segmentation column. Similar argument holds well when the first character has concavity and in such a situation the number of pixels decreases initially and after few columns it starts increasing. Whenever any downward trend appears it is marked as segmentation column. Second example word in Fig. 18 shows this concept.

Segmentation problem in this zone becomes difficult when *Kanoda* touches with other vowels. As the shape of this character contains one little concavity followed by one little convexity, it produces incorrect segmentation. But the chances of occurring touching characters involving this character are very less. Noise may also 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 shown in Fig. 19(c).

Fig. 18 a) example words, b) problem areas, c) top profile of problem areas, d) segmenting columns in top profiles, e) actual segmented characters

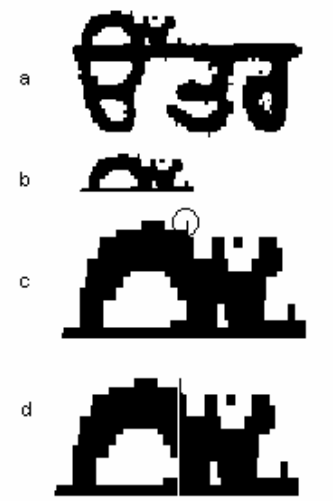

Fig. 19 a) Example word, b) problem area, c) extended view of problem area and noise pixel encircled, d) incorrect segmentation column

VIII. Segmentation in Lower Zone

As described in section V (D), touching characters in lower zone can be divided into two categories. For first category it is sufficient to identify the base line of a strip. Base line segments the middle zone characters from lower zone characters. For second kind of touching category, 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 fourth category in middle zone in section VI (D).

IX. Results and Discussions

The main objective of the work was to segment the multiple horizontally overlapping lines and to segment the touching characters present in all the three zones in degraded printed Gurmukhi script. For segmenting horizontally overlapping lines, we have scanned 54 documents from various printed newspapers articles. We applied algorithm 1 on these documents and achieved 95% accuracy in segmenting the horizontally overlapping lines along with associating the small

size strips (containing only lower/upper zone) to their respective text lines. Work is in progress to implement the same algorithm to segment the horizontally overlapping lines from various other Indian languages such as Devanagari, Bangla, Malayalam etc. We are also trying to work upon segmenting horizontally overlapping lines from newspaper articles, which contain characters of varying size and fonts.

For segmenting the touching characters in middle zone, we have implemented second algorithm. We collected about 6000 words containing touching characters and applied algorithm 2 on these words. The characters falling in category 4 were segmented using the solution scheme given section VI(D). The results of the implementations of these algorithms and solutions on nine Gurmukhi documents are given in Table III. The percentage accuracy of segmentation for these documents is in the range of 77-86%. Earlier [10], we have given higher accuracy, since the database taken was not very big at that time as compared to now.

TABLE III
ACCURACY IN MIDDLE ZONE

Document	Total characters	Number of touching characters in category				Correctly segmented	Wrong segmentation	% accuracy
		A	B	C	D			
Doc1	955	39	12	14	2	56	11	83.82
Doc2	1023	46	9	30	13	81	17	82.65
Doc3	984	105	26	21	18	131	39	77.05
Doc4	1023	119	33	41	25	173	45	79.35
Doc5	1045	125	34	36	27	189	33	85.13
Doc6	992	57	7	14	22	86	14	86
Doc7	923	41	6	17	6	58	12	82.85
Doc8	776	14	4	2	15	27	8	77.14

Similarly, for segmenting the touching characters in upper zone we have implemented Algorithm 3 proposed in section VII and observed that an accuracy of 76-82 has been achieved. The results are shown in Table IV.

TABLE IV
ACCURACY IN UPPER ZONE

Document	Total characters in upper zone	Touching characters	Correctly segmented characters	Wrong segmentation	% Accuracy
Doc1	307	19	15	4	78.94
Doc2	357	15	12	3	80.0
Doc3	319	22	18	4	81.81
Doc4	297	17	13	4	76.47

For segmenting the touching characters of first category in lower zone, the technique given in section VIII has been implemented. It has been observed that 92% of lower zone characters are correctly segmented from middle zone characters by using the concept of base line. We have also achieved 96% accuracy for segmenting the touching characters of second category in lower zone.

REFERENCES

[1] Y. Lu, "Machine Printed Character Segmentation – an Overview", *Pattern Recognition,* vol. 29, no. 1, pp. 67-80, 1995

[2] S.Kahan, T.Pavlidis, and H.S.Baird, " on the recognition of printed characters of any fonts and sizes", IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 9, no. 2, pp. 274-288, Mar. 1987

[3] S. Tsujimoto and H. Asada, " Resolving Ambiguity in Segmenting Touching Characters" Ist Int. Conf. on Document Analysis and Recognition ,pp. 701-709, Saint-Malo, France, Oct 1991.

[4] R.G.Casey and G. Nagy, "Recursive Segmentation and Classification of Composite character Patterns", Proc. 6th Int. Conf. on Pattern Recognition, pp. 1023-1026, Munich, germany,1982.

[5] Tao Hong, **"**Degraded text recognition using visual and linguistic context**",** *a dissertation submitted to the faculty of the graduate school of the State University of New York at Buffalo,* 1995.

[6] Veena Bansal and R.M.K. Sinha , "Segmentation of touching and Fused Devanagari characters, ", Pattern recognition, vol. 35, pp. 875-893, 2002.

[7] U. Garain, B.B. Chaudhuri, "Segmentation of touching characters in printed Devanagari and Bangla scripts using fuzzy multifactorial analysis", IEEE Trans. Systems Man Cybern. Part C-32 (2002) 449–459.

[8] B.B. Chaudhuri ,U. Pal and M. Mitra , "Automatic Recognition of Printed Oriya Script", *ICDAR,* pp.795-799,2001.

[9] U. Garain, B.B. Chaudhuri, "On recognition of touching characters in printed Bangla Documents", Proceedings of the Fourth International Conference on Document Analysis and Recognition, 1997, pp. 1011–1016.

[10] M. K. Jindal, G.S. Lehal and R.K. Sharma," A Study of Touching Characters in degraded Gurmukhi Script", in Int. Conf. on Pattern Recognition and Computer Vision, PRCV 2005, pp. ?, 25-27 February 2005, Istanbul, Turkey

[11] G. S .Lehal and Chandan Singh, "Text segmentation of machine printed Gurmukhi script", *Document Recognition and Retrieval VIII, Proceedings SPIE, USA, vol. 4307*, pp. 223-231, 2001.

[12] G. S. Lehal and Chandan Singh, "A technique for segmentation of Gurmukhi script", *Computer Analysis of Images and Patterns, Proceedings CAIP 2001,* W. Skarbek (Ed.), Lecture Notes in Computer Science, vol. 2124, Springer-Verlag, Germany, pp. 191-200, 2001.

[13] Serban, Rajjan and Raymund, "Proposed Heuristic Procedures to Preprocesses Character Pattern using Line Adjacency Graphs", *Pattern recognition,* vol. 29, no. 6, pp. 951-975, 1996.

[14] B. B. Chaudhuri and U. Pal**, "**Skew Angle Detection of Digitized Indian Scripts Documents**",** *Pattern recognition,* vol. 19, no. 2, pp. 182-186, 1997.