A Complete Machine printed Gurmukhi OCR System

G. S. Lehal[1] and Chandan Singh[2]
[1] Department of Computer Science and Applications, Punjabi University, Patiala, India
gslehal@mailcity.com
[2] Department of Computer Science and Applications, Punjabi University, Patiala, India.
chandan@pbi.ernet.in

Abstract. Recognition of Indian language scripts is a challenging problem. Work for the development of complete OCR systems for Indian language scripts is still in infancy. Complete OCR systems have recently been developed for Devanagri and Bangla scripts. Research in the field of recognition of Gurmukhi script faces major problems mainly related to the unique characteristics of the script like connectivity of characters on the headline, characters in a word present in both horizontal and vertical directions, two or more characters in a word having intersecting minimum bounding rectangles along horizontal direction, existence of a large set of visually similar character pairs, multi-component characters, touching characters which are present even in clean documents and horizontally overlapping text segments. This paper addresses the problems in the various stages of the development of a complete OCR for Gurmukhi script and discusses potential solutions. A multi-font Gurmukhi OCR for printed text with an accuracy of more than 97% at character level is presented.
Keywords. OCR**,** Gurmukhi, Segmentation, Classification, Post processing

1. Introduction

During the past fifty years, optical character recognition systems have come a long way from one-of-a-kind special purpose readers to the multi-purpose production and interactive on-line systems of today. This progress has lowered data capture costs and has caused development of more reliable and accurate OCR systems. Now, commercial OCR systems for Latin characters are widely available on personal computers. Further, systems in the market can now read a variety of writing styles (e.g., handwritten, printed omni-font) and character sets including Chinese, Japanese, Korean, Cyrillic, and Arabic. Modern OCR software is highly accurate, easy to use and affordable and for the first time OCR looks set to be adopted in all kinds of work environments on a mass scale.
Research on Devanagri, Tamil and Telugu optical text recognition started around mid 70s[1-4]. But the research had only theoretical importance and it did not lead to development of a practical OCR system. It was only around mid 90s that researchers started working for development of complete OCR systems for Indian scripts such as Devanagri and Bangla[5-6].

The research work on OCR of Gurmukhi script is in its infancy. Lehal and Singh[7] and Goyal et al[8] have presented segmentation schemes for Gurmukhi text. Lehal and Singh[9] have also developed feature extraction and classification schemes for machine recognition of Gurmukhi characters. A post processing system for an OCR of Gurmukhi script has also been presented by Lehal and Singh[10].

Gurmukhi script is used primarily for the Punjabi language, which is the world's 14th most widely spoken language. The populace speaking Punjabi is not only confined to North Indian states such as Punjab, Haryana, Delhi, Rajasthan and Jammu & Kashmir but is spread over all parts of the world. It is spoken by over 30 million people in India as well as people living in far flung countries such as UK, USA, Canada, UAE, Singapore, Kenya, Fiji and Malaysia. There is rich literature in this language in the form of scripture, books, poetry, etc. Gurmukhi is the first official script adopted by Punjab state. It is also the second language in many northern states of India. It is, therefore, important to develop OCR for such a rich and widely used language which may find many practical use in various areas. In this paper we present a complete OCR system for Gurmukhi script. To the best our knowledge, this is the first paper dealing with a complete OCR system for Gurmukhi script.

2. Characteristics of Gurmukhi Script

Gurmukhi script like most of other Indian language scripts is written in a nonlinear fashion. The width of the characters is also not constant. The vowels getting attached to the consonant are not in one (or horizontal) directions, they can be placed either on the top or at the bottom of consonants. This makes the use of the script on computers more complicated to represent and process. Some of the major characteristics of the Gurmukhi script from OCR point of view are:

- **Character set :** Gurmukhi script is syllabic in nature. Gurmukhi script-consists of 41 consonants called *vianjans*, 9 vowel symbols called *laga* or *matras*, 2 symbols for nasal sounds(ਂ , ੰ), one symbol for reduplication of sound of any consonant (ੱ) and three half characters (੍ਰ ੍ਵ ੍ਹ), which lie at the feet of consonants. The complete Gurmukhi character set is shown in Fig 1.The first three consonants (ੳ, ਅ, ੲ) are classified as open syllabals and called vowel consonants or semiconsonants or *Matra Vahak* due to their inherent property that they are never used in work without any *Laga* or 'Vowel'. The next two consonants are classified as root class consonants. The rest of the consonants except to the last two groups namely the - *Antim* and *Naveen* group, are categorized according to their phonetic structure. There are five such categories namely the *Kakvarg toli, Chachvarg toli, Ttatvarg toli, Tatvarg toli* and the *Pavarg toli* depending upon the different organs like throat, palate, mouth, tongue and lips, using which they are pronounced or from where they originate.The last but one group consisting of 5 independent consonants (ਯ, ਰ, ਲ, ਵ, ੜ) is called the *Antim* group and the last group (ਸ਼, ਖ਼, ਗ਼, ਜ਼, ਲ਼, ਫ਼) is the *Naveen* group which has been introduced to accommodate the words of Persian, Arabic and Sanskrit.

Consonants						
ੳ	ਅ	ੲ				*Matra Vahak*
		ਸ	ਹ			*Mul Varag*
ਕ	ਖ	ਗ	ਘ	ਙ		*Kakvarg Toli*
ਚ	ਛ	ਜ	ਝ	ਞ		*Chach Varg Toli*
ਟ	ਠ	ਡ	ਢ	ਣ		*Ttatvarg Toli*
ਤ	ਥ	ਦ	ਧ	ਨ		*Tatvarg Toli*
ਪ	ਫ	ਬ	ਭ	ਮ		*Pavarg Toli*
ਯ	ਰ	ਲ	ਵ	ੜ		*Antim Toli*
ਸ਼	ਜ਼	ਖ਼	ਫ਼	ਗ਼	ਲ਼	*Naveen Toli*
Vowels						
ਾ ਿ ੀ	ੁ	ੂ	ੇ ੈ	ੋ	ੌ	
Additional symbols						
ਂ ੰ ੱ						
Half Characters						
੍ਹ	੍ਰ	੍ਵ				

Fig 1 : Gurmukhi Character Set

- **Connectivity of symbols :** Most of the characters have a horizontal line at the upper part. The characters of a word are connected mostly by this line called head line and so there is no vertical inter-character gap in the letters of a word and formation of merged characters is a norm rather than an aberration in Gurmukhi script
- **Word Partitioning into zones :** A word in Gurmukhi script can be partitioned into three horizontal zones (Fig. 2). The upper zone denotes the region above the head line, where vowels reside, while the middle zone represents the area below the head line where the consonants and some sub-parts of vowels (ਾ ਿ ੀ)are present. The middle zone is the busiest zone. The lower zone represents the area below middle zone where the two vowels (ੁ ੂ) and half characters lie in the foot of consonants.
- **Multi component characters :** There are many multi-component characters in Gurmukhi script. A multi-component character is a character which can decompose into isolated parts (e.g. ਸ਼, ਖ਼, ਜ਼, ਗ਼, ਫ਼, ੂ)
- **Frequently touching characters :** Many of the characters in the lower zone of a text line frequently touch the characters in the middle zone. Upper zone characters are also occasionally merged into a single component.

- **Similarity of group of symbols:** There are a lot of topologically similar character pairs in Gurmukhi script. They can be categorized as
  1. Character pairs which after thinning or in noisy conditions appear very similar (ਟ and ਦ, ਤ and ੜ, ਬ and ਥ, ਤ and ਡ, ੑ and ੍)
  2. Character pairs which are differentiated whether or not they are open/closed along the headline (ਸ and ਮ, ਧ and ਪ, ਥ and ਖ)
  3. Character pairs which are exactly similar in shape but are distinguished only by the presence/absence of a dot in the feet of a character (ਸ and ਸ਼, ਖ and ਖ਼, ਜ and ਜ਼, ਫ and ਫ਼, ਗ and ਗ਼)

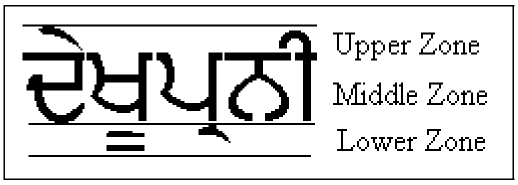


Fig 2 : Three zones of a word in Gurmukhi script

3. System Overview

The overall system design of the Gurmukhi OCR system developed and implemented is shown in Fig. 3. As with most of the OCR systems, there are five main processing stages: *Digitization, Pre-processing, Segmentation, Recognition* and *Post-processing.*

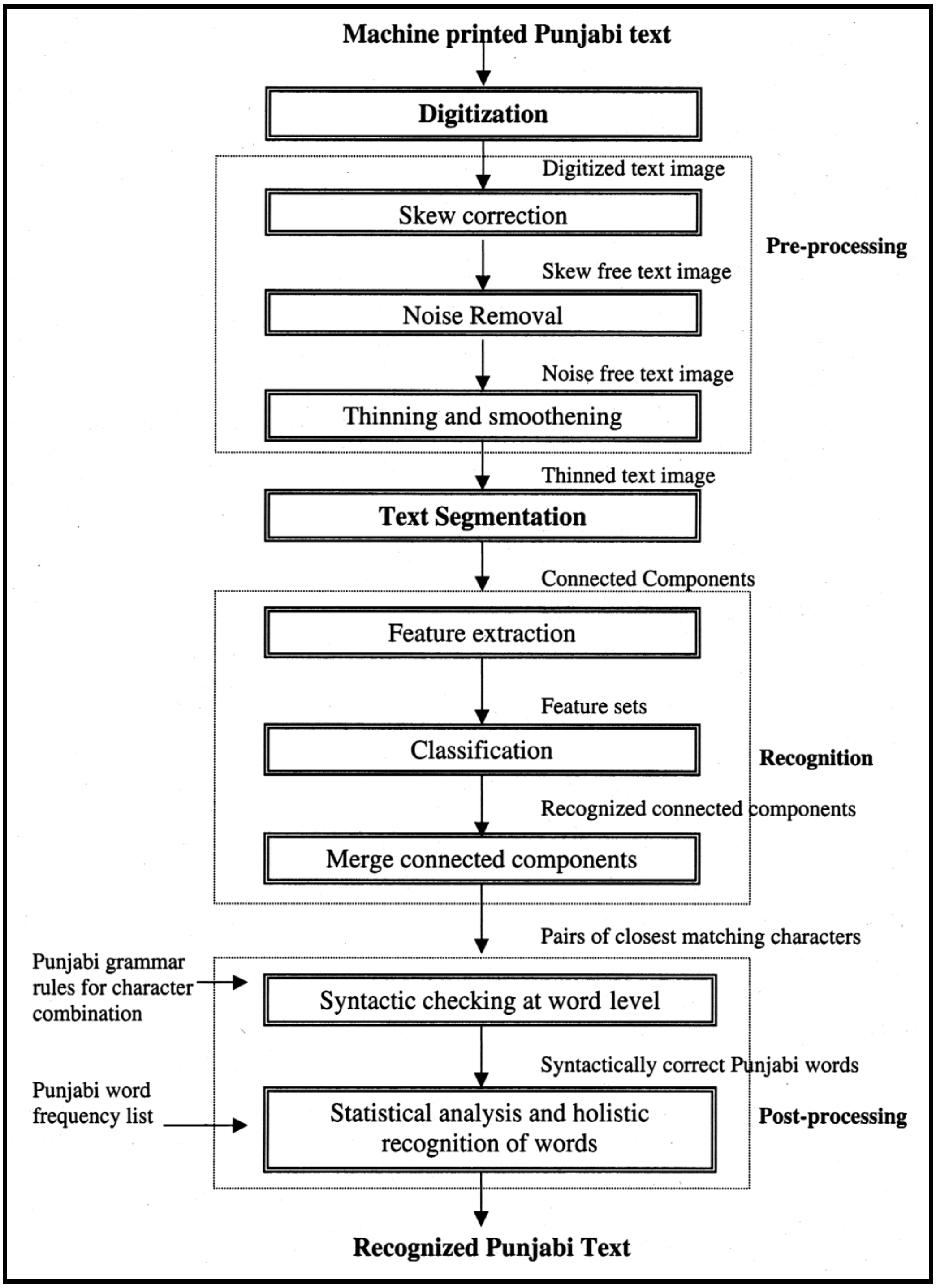
4. Digitization and Pre-processing

In order to recognize a text document, the first step consists of converting the document into a numerically representable form. The conversion process is physically accomplished by a digitizer, which can either be a scanner or a camera. The scanning resolution varies from 100 to 900 dots per inch (dpi). In our present work we have used a scanning resolution of 300 dpi.

4.1 Pre-Processing

The pre-processing stage is a collection of operations that are applied successively on an image. It takes in a raw image and improves it by reducing noise and distortion, removing skewness and skeltonizing the pattern. In our current work we have performed the following pre-processing steps:

1. Skew detection and correction
2. Noise removal
3. Thinning
4. Smoothening the headline


Machine printed Punjabi text
Digitization
Digitized text image
Skew correction
Pre-processing
Skew free text image
Noise Removal
Noise free text image
Thinning and smoothening
Thinned text image
Text Segmentation
Connected Components
Feature extraction
Feature sets
Classification
Recognition
Recognized connected components
Merge connected components
Pairs of closest matching characters
Punjabi grammar rules for character combination
Syntactic checking at word level
Syntactically correct Punjabi words
Punjabi word frequency list
Statistical analysis and holistic recognition of words
Post-processing
Recognized Punjabi Text
Fig. 3 : An over view of the Gurmukhi script recognition system

4.1.1 Skew Detection & Correction

Skewness refers to the tilt in the bitmapped image of the scanned paper for OCR. It is usually caused if the document is not well aligned on the scanner, thus yielding a skewed (rotated) digital image. The segmentation and feature extraction algorithms developed by the authors for the Gurmukhi OCR are sensitive to the orientation (or skew) of the input document image making it necessary to develop algorithms to perform skew detection and correction automatically. We have used the skew detection and correction technique for machine printed Gurmukhi script developed by Lehal and Dhir[11]. An advantage of this technique is that it is not constrained to any range and works correctly in the presence of graphics and tables in the text image.
The algorithm works in three stages. The skew angle is determined by calculating horizontal and vertical projections at different angles at fixed interval in range [0°, 90°]. Under such projections, for an image with no skew, headlines appear as distinct peaks while gaps between successive text rows will be represented by valleys. The task is to determine the angle at which the highest peaks and deepest valleys in the projections are present. Both the horizontal and vertical profiles are simultaneously examined for peaks and valleys. In order to decrease the computational cost, first a rough estimate of the skew angle is made by taking the angle interval 3°. Once this estimate is calculated, the accurate skew angle θ is determined by looking in the range [θ - 3°, θ + 3°] at an interval of 0.25°. The image is then rotated over -θ, where θ is the skew angle. Since the skew angle is checked only in the range [0° - 90°] and the image can be skewed at any angle in the range [-180°, 180°], the rotated image may need another additional rotation by 90°, -90° or 180°, depending on the skewness of the image. After first rotation the bitmap image will be aligned along x or y-axis. If the rotated image is skewed at 90° or -90°, then the highest peaks and valleys would be present in vertical projection else they will be reported in horizontal projection. The physical characteristics of the Gurmukhi script are then used to determine the skew angle of the image after first rotation. To determine the skew angle of the image aligned with y-axis, if the black pixel density on the left side of headlines is greater than the black pixel density on right side of text rows then the image is skewed at -90° else it is skewed at 90°. Similarly for the image aligned with x-axis, if the black pixel density above the headlines is lesser than the black pixel density below the headline then the image is straight else it is upside down. The image is rotated by the second rotation angle to completely remove any skewness present in the image.

4.1.2 Noise Removal

A preliminary noise removal algorithm has been employed to remove isolated black pixels and fill up the gaps in image regions by examining the 3x3 neighbourhood of white pixels. In case the number of black pixels in the 3x3 neighbourhood is more than six, then the white pixel is converted to black.

4.1.3 Thinning

Thinning is an essential step for many structural feature extraction methods. It reduces patterns to their skeletons or single pixel width pattern. It is often an efficient method for expressing structural relationships in characters as it reduces space and processing time by simplifying data structures. However, thinning has some disadvantages too. Thinning is a time-consuming process which may remove structurally significant portions of the image such as short protrusions, or introduce extraneous limbs or 'hairs'. In our present work, segmentation and feature extraction stages have greatly been simplified by working on thinned images of text, though in some of the cases the character shapes were slightly deformed. After experimenting with some of the common thinning algorithms, we have settled for the thinning algorithm suggested by Abdulla et al[12] for skeletonizing the Gurmukhi text images, as it was found to be the most suitable.

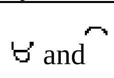
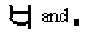
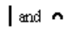
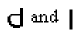
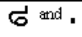
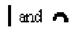
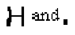
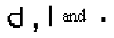
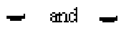
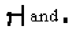
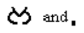
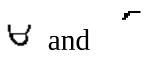
5. Text Segmentation

Gurmukhi script is a two dimensional composition of consonants, vowels and half characters which require segmentation in vertical as well in horizontal directions. Thus the segmentation of Gurmukhi text calls for a 2D analysis instead of the commonly used one-dimensional analysis for Roman script. In addition to the common segmentation problems faced in Indian language scripts, Gurmukhi script has other typical problems such as horizontally overlapping text segments and touching characters in various zonal positions in a word.

Since it is difficult to separate a cursive word directly into characters, a smaller unit than a character is preferred. To simplify character segmentation in our current work, we have taken an 8-connected component as the basic image representation throughout the recognition process and thus instead of character segmentation we have performed *connected component segmentation*. The segmentation stage breaks up a word and characters which lie above and below the headline into connected components and the classifier has been trained to recognize these connected components or sub-symbols. Table 1 lists all the connected components or sub-symbols derived from the Gurmukhi characters. It is to be noted that the headline is not considered a part of the connected component.

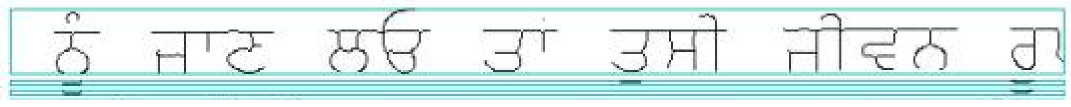
A combination of statistical analysis of text height and width, horizontal projection and vertical projection and connected component analysis are performed to segment the text image into connected components. We have employed a 5 phased segmentation scheme. These phases, which are described in detail in [7], are:

Table 1. Sub-symbols of Gurmukhi script used for segmentation and recognition

Symbol	**Sub-symbols**	**Symbol**	**Sub-symbols**	**Symbol**	**Sub-symbols**
ਉ		ਖ਼		ੀ	
ਗ		ਫ਼		ਿ	
ਸ਼		ਗ਼		=	
ਜ਼		ਲ਼		ਓ	
Gurmukhi Characters in upper zone	Same shapes retained	Gurmukhi Characters in lower zone	Same shapes retained	Rest of Gurmukhi characters in middle zone	Gurmukhi characters with their headlines stripped off

1. Dissect the text image into text strips using valleys in the horizontal projection profiles. Each of these strips could represent either one text row or only the upper or lower zones of a text row or more than one text row. For example, in Fig. 4a, one text row has been split into three strips after the application of horizontal projection. The first strip contains the characters present in upper and in middle zones of words while the next two strips contain the components of lower zone characters of above lying text row. Similarly the strip in Fig. 4b includes two text rows. These rows are horizontally overlapping and as such cannot be separated by horizontal projection profile.
2. Perform statistical analysis to automatically label the text strips as multi strip, core strip, upper strip or lower strip, depending on whether the text strip contains more than one text row, one text row, upper zone or lower zone of a text row respectively. For example, in Fig. 5, strip nos. 2 and 3 are lower strips, strip no. 1 is the core strip, strip no. 12 is the upper strip and strip no. 15 is the multi strip.
3. Decompose the text strips into smaller components such as words and connected components using vertical projection profile analysis. In case of multi strip, the strip is first split into individual text rows using the statistics based on the average height of a core strip and next each text row is split into words. In case of upper and lower strips there are no words and we just have sub parts of upper and lower zone vowels respectively. A connected component analysis is carried out to obtain the connected components in these strips.

4. Split words into connected components in case of core strip and multi strip. For obtaining the connected components the headline is rubbed off and after segmentation it is restored back.
5. Detect and segment touching characters in connected components. This phase is explained briefly in the following subsection.


(a)

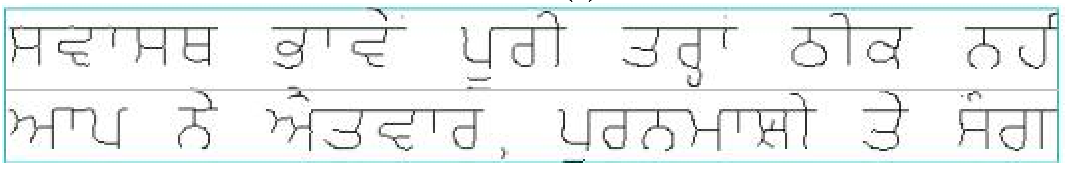

(b)

Fig 4 : a) Text row split into three text strips b)Text strip containing two text rows. Gray line indicates the overlap region.

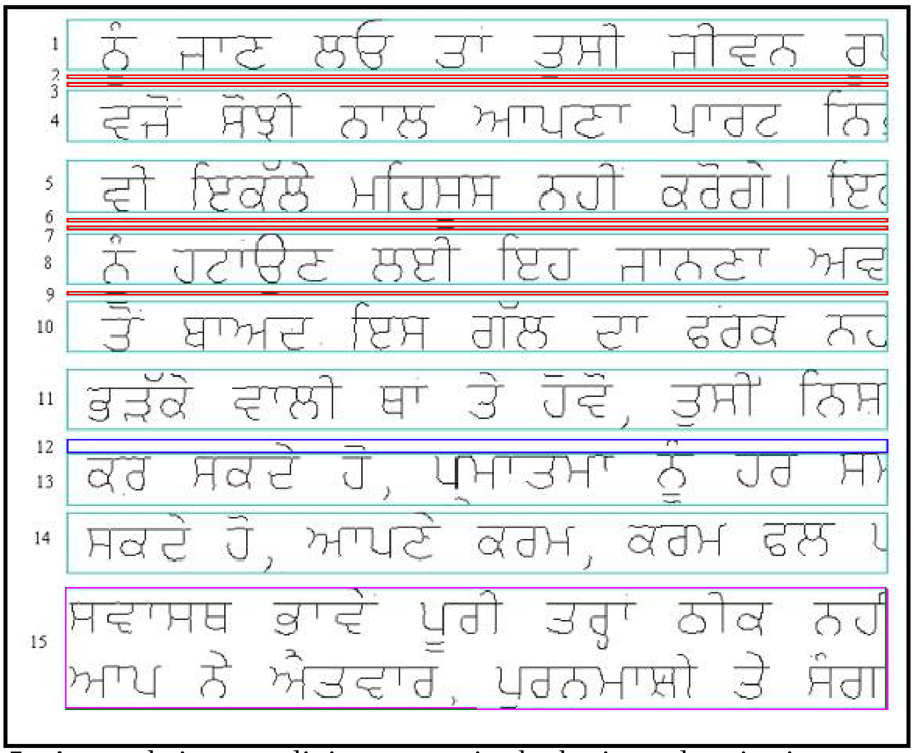


Fig. 5. A sample image split into text strips by horizontal projection

5.1 Touching Characters

It has been observed that touching characters are frequently present even in clean machine printed texts. As already mentioned, segmentation process for Gurmukhi script proceeds in both x and y directions, since two or more characters of a word may be sharing the same x coordinate. Therefore, for the segmentation of touching characters in Gurmukhi script, the merging points of the touching characters have to be determined both along the x and y axes. These touching characters can be categorized as follows:

a) Touching characters in upper zone
b) Touching characters in middle zone
c) Lower zone characters touching with middle zone characters
d) Lower zone characters touching with each other

Fig. 6 shows examples of touching characters for these categories. The statistics such as average character width and height and certain heuristics were developed to solve the segmentation problem for Gurmukhi characters. The details are discussed elsewhere[7]. It was found during experiments that 6.9% of upper zone, 0.12% of middle zone characters, 19.11% of lower zone and middle zone characters and 0.03% of lower zone characters were touching with each other.

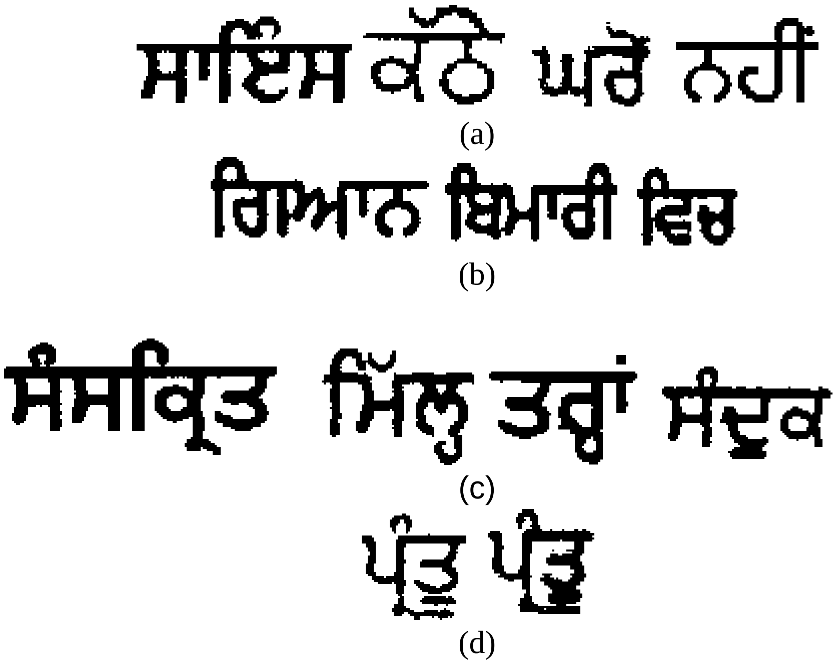


Fig. 6. Examples of touching characters a) touching characters in upper zone, b)touching characters in middle zone, c) Lower zone characters touching with middle zone characters, d) Lower zone characters touching with each other

6. Recognition Stage

The main phases of the recognition stage of OCR of Gurmukhi characters in our present work are:

1) Feature extraction
2) Classification of connected components using extracted features and zonal information.
3) Combining and converting the connected components to form Gurmukhi symbols.

6.1 Feature Extraction

After a careful analysis of shape of Gurmukhi characters for different fonts and sizes, two sets of features were developed. The first feature set called primary feature set is made up of robust and font and size invariant features. The purpose of primary feature set is to precisely divide the set of characters lying in middle zone into smaller subsets which can be easily managed. The cardinality of these subsets varies from 1 to 8. The Boolean valued features used in the primary feature set are:

1) **Number of junctions with the headline (P_1) :** It can be noted that each character in Gurmukhi has either 1 or more than 1 junctions with the headline. For example, the character ਫ has one junction while ਪ has 2 junctions. This feature has been used to divide the complete Gurmukhi character set into almost 2 equal sized subsets. This feature is true if the number of junctions is 1 else it is false.
2) **Presence of sidebar (P_2) :** The presence or absence of sidebar is another very robust feature for classifying the characters. For example ਮ, ਯ and ਹ have a sidebar while ਕ, ਙ and ਛ do not have it. This feature is true if a vertical line is present on the rightmost side of the sub-symbol else it is false.
3) **Presence of a loop (P_3) :** The presence of a loop in the sub-symbol is another important classification feature. The loop should not be formed along the headline. Thus this feature is true for sub-symbol of ਹ but is false for sub-symbol of ਯ since headline is involved in the loop.
4) **No Loop formed with headline(P_4) :** This feature is true if the character is open at top along the headline or in other words if there is no loop containing headline as its subpart. Examples of characters with this feature are ਹ and ਖ while it is absent in ਲ and ਯ.

The second feature set, called secondary feature set, is a combination of local and global features, which are aimed to capture the geometrical and topological features of the characters and efficiently distinguish and identify the character from a small subset of characters. The secondary feature set is used for classification of all the characters of the Gurmukhi script lying in any one of the three zones, while primary feature set is used only for middle zone characters.
The Secondary Feature Set consists of following features:

1) **Number of endpoints and their location (S_1)** : A black pixel is considered to be an end point if there is only one black pixel in its 3 x 3 neighborhood in the resolution of the character image. In order to determine the position of an endpoint in one of the 9 quadrants, the character image is divided into a 3x3 equal zones that are numbered 1 through 9(Fig. 7). Using these zones, the position of the endpoints in terms of their positions in quadrants and their numbers are noted.

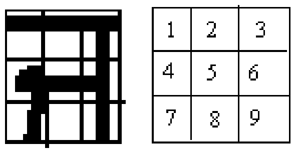


Fig. 7 : A character image divided into a 3x3 equal zones

2) **Number of junctions and their location (S_2)**: A black pixel is considered to be a junction if there are more than two black pixels in its 3 x 3 neighbourhood in the resolution of the character image. The number of junctions as well as their positions in terms of 9(3x3) quadrants are recorded. Junctions lying within a pre-defined radial distance are merged into a single junction and the junctions associated with the headline are ignored.
3) **Horizontal Projection Count (S_3)**: Horizontal Projection Count is represented as $HPC(i) = \sum_j F(i, j)$, where F(i,j) is a pixel value (0 for background and 1 for foreground) of a character image, and i and j denote row and column positions of a pixel, with the image's top left corner set to F(0,0). It is calculated by scanning the image row-wise and finding the sum of foreground pixels in each row. To take care of variations in character sizes, the horizontal projection count of a character image is represented by percentage instead of an absolute value and in our present work it is stored as a 4 component vector where the four components represent the percentage of rows with 1 pixel, 2 pixels, 3 pixels and more than 3 pixels.

Left and Right Projection profiles (S_4 through S_8) : The next 5 features are based on projection profiles. Left projection of a character is derived by scanning each line of the character from top to bottom and from left to right, and by storing the first black pixel of the character in each row. Similarly the right projection profile is found by scanning the character from top to bottom and from right to left. The pixels lying along the headline are ignored while deriving the projection profiles.

4) **Right Profile depth (S_4)**: The maximum depth of the right profile is stored as percentage with respect to total width of the box enclosing the character image.
5) **Left Profile Upper Depth (S_5)**: The profile is computed from the left and the maximum depth of the upper half of the profile is stored as percentage with respect to total width of the box enclosing the character image.
6) **Left Profile Lower Depth (S_6)**: The maximum depth of the lower half of the left profile is stored as percentage with respect to total width of the box enclosing the character image.
7) **Left and Right Profile Direction Code (S_7, S_8)**: A variation of chain encoding is used on left and right profiles. The profile is scanned from top to bottom and local directions of the profile at each pixel are noted.

Starting from current pixel, the pixel distance of the next pixel in west, south or east directions is noted. The cumulative count of movement in the three directions is represented by the percentage occurrences with respect to the total number of pixel movement and stored as a 3 component vector with the three components representing the distance covered in west, south and east directions respectively. Thus if the movements in west, south and east directions are 4, 2 and 5 pixels respectively, then the direction code of the profile will be [37, 18, 45].

8) **Aspect Ratio (S_9) :** Aspect ratio which is obtained by dividing the sub-symbol height by its width, was found to be very useful for classifying the sub-symbols lying in lower-zone.
9) **Distribution of black pixels about the horizontal mid line (S_{10}) :** It has been found during experiments that some of the very closely resembling characters such as ਦ and ਟ and ੇ and ੋ were difficult to recognize specially after thinning. Most of their features are similar. So a new feature was introduced specifically for these character pairs, which was based on distribution of black pixels in each column along the horizontal mid line. This distance is calculated by moving from left to right and at each column determining the distance of the nearest black pixel in that column from the horizontal middle line. This distance is then summed and normalized by dividing with the area of the character image and then converted into percentage. To take care of the distortions at the endpoints in some of the character images, we ignore ten percent of vertical regions at both the ends. The character image area is the product of its height and the truncated width.

To further clarify the primary and secondary features used in our present study, we consider the three character images of Fig. 8. These images have been preprocessed and segmented from the text. The values of all primary and secondary features as calculated by the feature extractor are tabulated in Table 2. The values in brackets for features S_1 and S_2 represent the quadrant number. Thus 1(9) for feature S_2 means that there is one end point present in the character image in 9th quadrant.

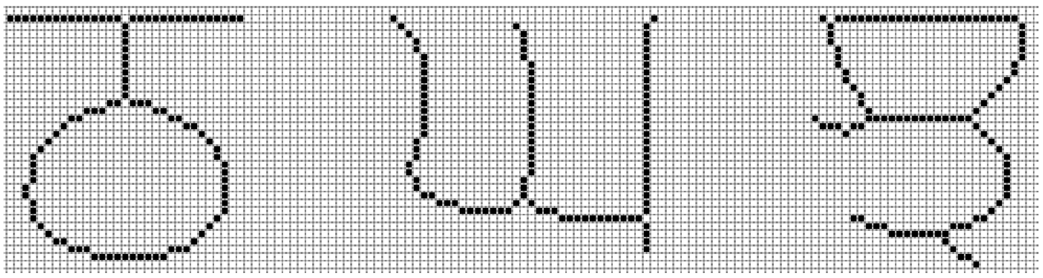


Fig 8 : Thinned character images

Table 2: Calculated values of primary secondary features for images of Fig. 8

Character → Feature ↓	ਠ	ਘ	ਝ
P_1	True	False	False
P_2	False	True	False
P_3	True	False	False
P_4	False	False	True
S_1	0	1 (9)	3(4, 7, 9)
S_2	1(2)	2(8, 9)	3(4, 6, 8)
S_3	[32, 45, 6, 16]	[13, 0, 73, 13]	[40, 43, 0, 15]
S_4	50	3	37
S_5	50	9	92
S_6	34	100	92
S_7	[35, 40, 24]	[4, 33, 62]	[32, 13, 54]
S_8	[21, 43, 35]	[3, 93, 3]	[42, 31, 25]
S_9	1.14	0.91	1.14
S_{10}	10	27	10

6.2 Classification

In our present work, we have used a multi-stage classification in which the binary tree and nearest neighbour classifiers have been used in a hierarchical fashion. The complete feature set used for classification using nearest neighbour classifier is tabulated in Table 3. This classification scheme for the Gurmukhi characters proceeds in 3 stages. These stages are:

1) Using zonal information, we classify the symbol into one of the three sets, lying either in upper zone, middle zone or in lower zone.
2) If the symbol is in the middle zone, then we assign it to one of the sets 1 to10 of Table 3 using primary features and binary classifier tree. At the end of this stage the symbol will be classified into one of 12 sets including the sets for characters in upper and lower zones.
3) Lastly, the symbol classified to one of the 12 sets of Table 3 is recognized using nearest neighbour classifier and the feature set of secondary features assigned for that particular set. It is to be noted that not all the secondary features are used for classification. Depending on the sub-set and the characters which have to be distinguished in that sub-set, we choose the features which are enough to effectively distinguish the characters of a sub-set. Thus, for example, for sub-set 6, which contains only two characters (ਥ ਬ), we note that the characters are quite close in appearance. They have same number of endpoints and joints as well as same features on the left profile. The only difference found is in the right projection profiles and so the secondary features S_5 and S_8 which are related to right profile are used for classification.

Table 3 : Secondary feature set for classification of character sets

Set #	**Character Set**	**Features for classification**
1	ਹ ਰ	S_1 S_2 S_3

2	ਹ ਜ ।	S_1 S_2 S_3
3	ਕ ਙ ਛ ਠ ਡ ਢ ਫ ਭ	S_1 S_2 S_3 S_4 S_5 S_6 S_7 S_8
4	ਟ ਣ ਤ ਦ ਨ ਵ ੜ	S_1 S_2 S_3 S_4 S_5 S_6 S_7 S_8 S_{10}
5	ਖ	-
6	ਥ ਬ	S_5 S_8
7	ਅ ਘ ਪ ਮ	S_1 S_2 S_3 S_5
8	ਸ ਧ ਯ	S_1 S_2 S_3 S_5
9	ਉ	-
10	ੲ ਝ ਞ ਲ	S_1 S_2 S_3 S_4 S_7 S_8
11	ਿ ੀ ੇ ੈ ੋ ੰ ਂ	S_1 S_7 S_8 S_{10}
12	੍ ੁ ੂ	S_8 S_9

6.2.1 Design of the Binary Tree Classifier

We have designed a strictly binary decision tree with 10 leaf and 9 non-leaf nodes. The leaf nodes correspond to the classification of the character in one of the 10 sub-classes. The height of the tree is 4. Only one feature is tested at each non-terminal node for traversing the tree. The decision rules are binary i.e. the presence/absence of the feature. The features at the non-terminal nodes are chosen according to their robustness and tolerance to noise and remain invariant under font and image size. The most stable feature is used at root node and it divides the character set into two almost equal sized subsets. The complete binary tree classier is shown in Fig. 9.

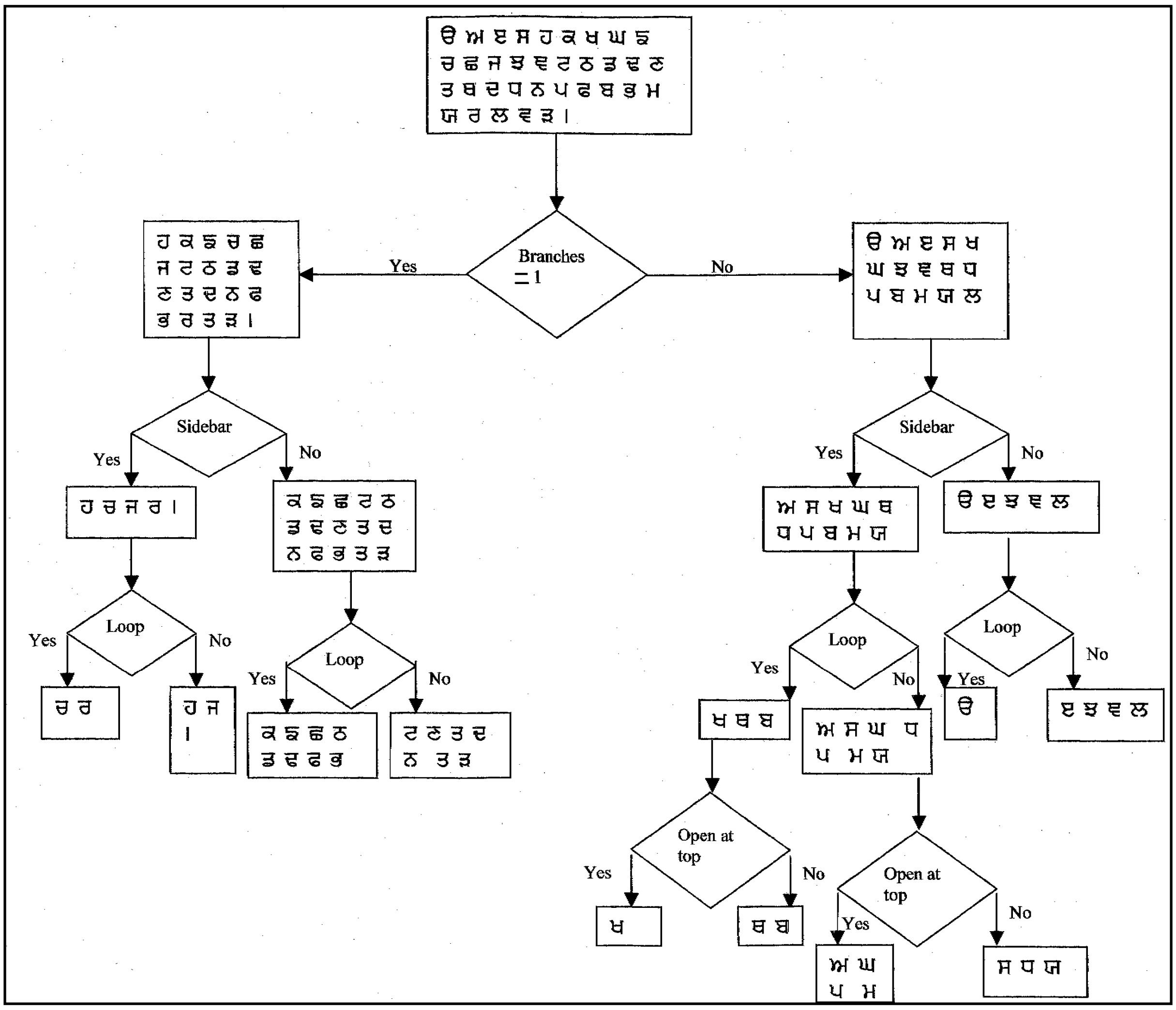


Fig. 9 : Binary Classifier tree

6.3 Merging Sub-symbols

In this last stage of recognition of characters, the information about coordinates of bounding box of sub-symbols and context is used to merge and convert the sub-symbols to Gurmukhi characters. It is to be noted that most of the sub-symbols can be converted as such to equivalent character (Table 1). It is only in some typical cases where a character may be broken into more than one sub-symbol that some rules have to be devised to merge these sub-symbols. For example, if the sub-symbol in middle zone is ਰ and the next sub-symbols in middle and upper zones are . and ਿrespectively, then if the upper sub-symbol is vertically overlapping with one or more middle zone sub-symbols, then these sub-symbols might represent one of the character combinations ਰੀ, ਰਿ or ਗੰ.The information regarding the overlapping of the upper and middle zone connected components (CCs) is used to identify the characters represented by the CCs. Thus, if ◠ is overlapping with both ਂ and ı then the CCs combine to form ਰੀ . If ◠ is overlapping with only ı then the CCs combine to form ਰਿ and if ◠ is overlapping only with only ਂ then the CCs combine to form ਰੀਂ.

7. Post Processing

In order to rectify the classification errors, the output of classification stage is fed to the post processor. For the post processing we have used a Punjabi corpus, which serves the dual purpose of providing data for statistical analysis of Punjabi language and also for checking the spelling of a word. Punjabi grammar rules are also incorporated to check for illegal character combinations such as presence of two consecutive vowels or a word starting with a forbidden consonant or vowel.
A word frequency list is created from the Punjabi corpus. The list stores the frequency of occurrence of all words present in the corpus. The list is then partitioned into smaller sub lists based on the word size. We have created 7 sub-lists corresponding to word sizes of two, three, four, five, six, seven and greater than seven characters. Further, in each of this sub-list, a list of visually similar words is generated. We say that two words are visually similar, if each character in the corresponding position of the two words is visually similar. To decide the visual similarity of two characters, the zonal position of the character and the primary features, discussed in section 6.1 are used. The Gurmukhi character set is divided into 16 sub-sets consisting of visually similar characters. Out of the 16 sub-sets, the first ten sub-sets(0-9) contain the characters present in the middle zone. The middle zone characters are categorized using the four Boolean valued primary features, which have been described in section 6.1. These features, as already discussed, are very robust and easy to detect and need not be computed again as they are available from the recognition stage. All the members of a sub-set share the same Boolean values of the primary features. For example, for all the members of sub-set no. 2 (Table 4), the value of the first feature is true, second feature is false, third feature is true and fourth feature is false, since all the characters in this sub-set have one branch from the headline, do not have a side bar, contain a loop but no loop is formed along the headline.

The eleventh and twelfth character sub-sets correspond to the upper zone and lower zone characters respectively. We have created separate sub-sets for some of the most frequently occurring characters, which have a very high recognition rate and are not confused with any other character. The thirteenth sub-set contains only the character ਾ. From a statistical analysis of the corpus it was found that the character ਾ is the most frequently occurring character with a frequency of occurrence of 10% and is very easily detectable. Similarly the character ਂ , which is just a dot present in the upper zone and hereby referred as *bindi*, is very easily recognizable and not confused with any other character and has 5% frequency of occurrence, is assigned the fourteenth sub-set. The fifteenth and sixteenth character sub-sets consist of ਿ and ੀ characters. These characters, which are present in both upper and lower zones, have a high frequency of occurrence, and have no confusion with any other character. The complete sub-sets are shown in Table 4.

Table 4 : Partitioning of Gurmukhi character set into 16 sub-sets

Sub-set No.	Character Set	Sub-set No.	Character Set
0	ਚ ਰ	8	ੳ ਓ
1	ਹ ਜ ਜ਼	9	ੲ ਝ ਵ ਲ
2	ਕ ਙ ਛ ਠ ਡ ਢ ਫ ਭ ਫ਼	10	ੇ ੈ ੋ ੌ ੂ ੑ
3	ਟ ਣ ਤ ਦ ਨ ਵ ੜ	11	ੁ ੂ ੍ ੵ ੑ
4	ਖ ਖ਼ ਗ ਗ਼	12	ਾ
5	ਥ ਬ	13	ਃ
6	ਅ ਘ ਪ ਮ	14	ਿ
7	ਸ ਧ ਯ ਸ਼	15	ੀ

The sub-lists are stored in arrays of structures of visually similar words. We generate seven such arrays for word sizes of two, three, four, five, six, seven and greater than seven characters. Each element in this array stores the information about the relative percentage frequency of occurrence of the visually similar words as well as the percentage frequency of occurrence of characters in different positions of the word. We call this array SSSL(Shape based Statistical Structure List). Each element of the array is assigned a unique code generated from the sub-list number of the characters and the array elements are arranged in sorted order of the code for faster searching. The structure of an element of the array SSSL is:

```
struct SSSL_element
{
  int code;
  struct char_freq_list *char_list;
  struct word_freq_list *word_list;
};
struct char_freq_list
{
   char punjabi_char;
   int frequeny;
   struct char_freq_list *next;
};

struct word_freq_list
{
   char *punjabi_word;
   int frequeny;
   struct word_freq_list *next;
};
```

As is clear from the structure, each element of SSSL has links to char_freq_list and word_freq_list. The structure, char_freq_list is a single linked list storing the percentage frequency of occurrence of characters at a particular position and word_freq_list is a singly linked of visually similar words storing their relative percentage of frequency. The header node of char_freq_list has pointers to next node of the char_freq_list and char_freq_list for next character position. For example, consider an eight word frequency list of words of length 3 (Table 5). From this frequency list, the SSSL, with two elements as shown in Fig. 10, is generated.

Table 6: A word frequency list of word size 3

Word	**Frequency**	**Word**	**Frequency**
ਜੋਰ	1400	ਹੋਰ	2600
ਹੇਰ	500	ਜੱਚ	1500
ਠੱਕ	2500	ਵੰਡ	4700
ਦੰਡ	1600	ਠੋਕ	1200

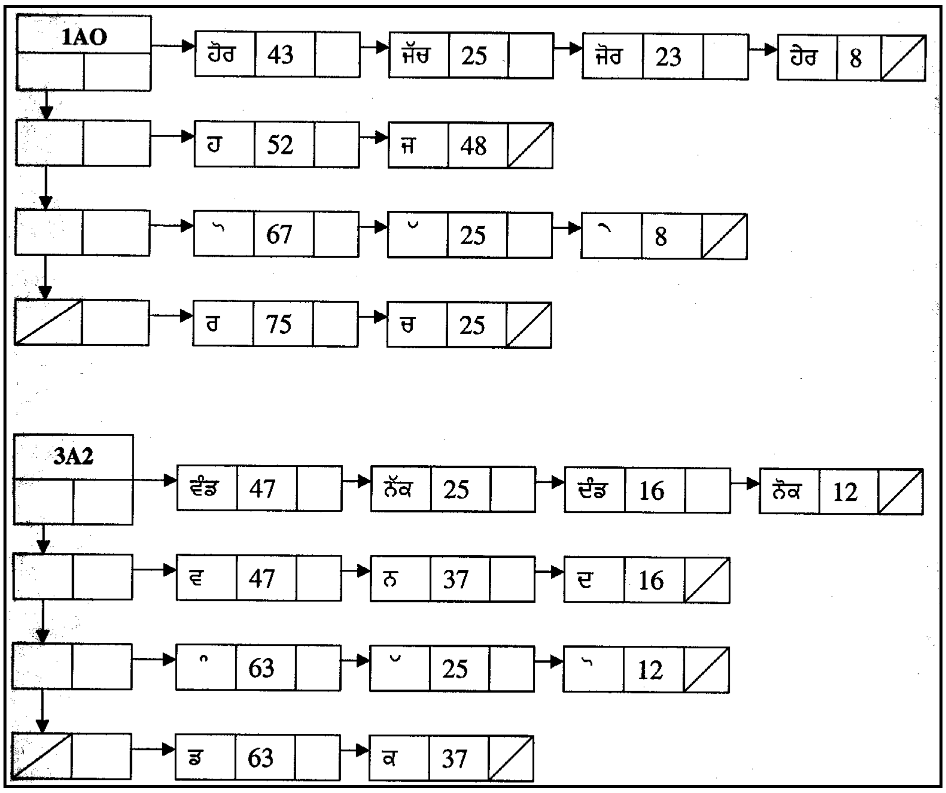


Fig 10 : SSSL generated from word frequency list of Table 5

The first element of the list in Fig 10 contains first four words of the frequency list. These words have visually similar characters in all the three positions and hence they are considered to be visually similar. The first character in each of these words belongs to sub-set 1, second character belongs to sub-set 10 and third character belongs to sub-set zero. Thus the code of this structure in hexadecimal format is 1A0, representing the subset of the three character positions. The character ਹ is present in the first position in the first and second word, giving a total frequency 3100 and similarly the character ਸ is present in first position in the third and fourth word, giving a total frequency 2900. The percentage frequency of occurrence of ਹ and ਸ in the first position is thus 52 and 48 respectively and these values are stored in the char_freq_list for first position of the word. Similarly the percentage frequency of occurrence of characters in second and third position is calculated and the char_freq_lists for those positions are generated. Similarly the relative percentage frequency of occurrence of the words is stored in the word_freq_list.

The classifier generates a character distance pair set (CD){(c_1, d_1), (c_2, d_2)}, where c_1 represents the best matching character and d_1 represents the distance of the input character image from the character prototype stored in the training data. Similarly c_2 and d_2 represent the second nearest matching character and the distance of the input character image with the character prototype. The CD pairs are stored for each character position in the word. The SSSL is combined with the CD pairs to predict the most likely character. The final decision of the choice of character is obtained by combining the results of the recognizer and the post processor. Depending on how closely the character image matches with the nearest character prototype and the second nearest character prototype, a decision is made on how much weightage has to be assigned to the post processor and the recognizer. We have used two weights, w_1 and w_2, where w_1 is based on the distance of the character with its nearest matching prototype(d_1) and represents the confidence of the recognizer. Smaller value of w_1 means that the character image is closely matching the library image and lesser weight has to be assigned to the post processor. The weight w_2 represents the closeness of the shapes of the top two choices. Higher value of w_2 means that the shape of the character images of top two choices are closely matching and the confusion is to be resolved by assigning more weight to the post processor. These weights are combined with the percentage of occurrence of the top two choices at a particular position in the word and the distance of top two choices with their nearest matching training set prototypes. The decision on the choice of one of the characters is taken as follows:
We calculate a parameter, dist, which represents the distance of the recognized character from the actual character. This has been formulated empirically as

$$dist = ((w_1+w_2)/(100.0))*(p_2-p_1) - (d_2-d_1)^2 \qquad (1)$$

where d_1 = distance of first choice ($char_1$) with the nearest matching library prototype

d_2 = distance of second choice ($char_2$) with the nearest matching library prototype

$w_1 = d_1^2$ subject to maximum of 50;

$w_2 = 50-(d_2 - d_1)^2$ subject to minimum of 0;

p_1 = percentage frequency of occurrence of $char_1$ in char_freq_list;

p_2 = percentage frequency of occurrence of $char_2$ in char_freq_list;

We recognize a character as $char_2$ if dist > 0 else it is retained as $char_1$

Using this technique we have been able to rectify more than one wrongly recognized character in a word. As an example, consider the skeletonized image in Fig. 11. The image in Fig. 11 represents the word ਬੋਲ but it has been identified as ਥੇਲ by the recognizer.

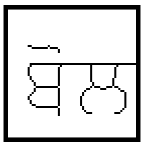


Fig 11 : A Sample image

The CD pairs generated for the first two wrongly identified characters of the word are {(ਥ ,1) , (ਬ,2)} and {(ੇ ,4) , (ੋ,6)}. The percentage frequency of occurrence of ਥ and ਬ in the first position in the corresponding char_freq_list is 0 and 100 respectively. Thus d_1=1,d_2=2, p_1=0, p_2=100, w_1=1, w_2=49 and dist=49. For second character d_1=4, d_2=6, p_1=1 and p_2=99, w_1=16 and w_2=46 and dist=56.Since dist is positive for both the cases so the second choice character is taken for both cases and thus the word is corrected as ਬੋਲ

The purpose of the word_freq_list is two fold:
Check for the existence of a word in the corpus: The recognized word is checked for its presence in the corpus by looking up the word_freq_list. If the word is not present then it is replaced with the nearest matching word provided that the distance of the recognized characters from the stored templates is greater than some preset threshold value. This is necessary to prevent accidental conversion of non-dictionary words such as proper nouns and abbreviations to a dictionary word.
Perform holistic recognition of a word: The words in the list are sorted in descending order of frequency of occurrence. If it is found that the first word in the list has frequency of occurrence greater than 90, then the recognized word, even though it may be present in the corpus, is converted to the high probability word again subject to the condition that the distance of the recognized characters from the stored templates is greater than some threshold value.

The Punjabi grammar rules are also used to eliminate illegal character combinations.

8. Experimental Results

We tested our OCR on 25 Gurmukhi text documents consisting of about 30000 characters. The documents were pages from good quality books and laser print outs in multiple sizes and fonts. We tested on font sizes in the range 10-24 point size and 8 fonts were used.
It was found that seven characters (ਹ ਖ ਘ ਙ ਞ ਢ ਯ) with a combined frequency of occurrences of 5.67% were recognized with almost 100%

accuracy. Out of these the character ਹ has a high frequency of occurrence (4.2%) but in the subset 2 (Table 2), there are only two other characters for resolving the confusion and their shapes are quite different so ਹ is not confused with them. Twenty two characters with cumulative frequency of occurrences of 44.69% are recognized with more than 98% accuracy. On the lower end, eleven characters (ਲ਼, ਜ਼ ੈ ਂ ਸ਼ ਥ ਚ ੂ ਖ਼ ਫ਼ ਗ਼) with a cumulative frequency of occurrences of 10.08% have a low recognition rate of 80% or less. It is these characters which are the main bottlenecks in the performance of the OCR. It can be seen that majority of these characters are the characters with dot at their feet. The reason for this inaccuracy is that during the thinning either the dot is deleted or it gets merged with the character. Even among the characters with dot at their feet the characters ਗ਼ ਖ਼ and ਫ਼ have a far more poor recognition accuracy as compared to characters ਲ਼ ਸ਼ and ਜ਼. The reason for this is that the dot is positioned in centre for characters ਲ਼ ਸ਼ and ਜ਼ while for characters ਗ਼ ਖ਼ and ਫ਼ the dot is positioned very close to the character and so it gets easily merged on thinning. The characters ਚ and ਥ have low recognition accuracy as they are very closely resembling with characters ਰ and ਧ respectively and are often confused with them. The characters ੈ and ੂ, have their strokes often joined together or touching with other characters which makes it difficult to recognize them. The character, :(*bindi*), which is similar to a dot and is present in the upper zone is also difficult to recognize. There were two type of errors produced: a) Deletion - The character *bindi* would be removed during the scanning and binarization process or by the thinning algorithm. In many cases the *bindi* character would be merged with other symbols in the upper zone and vanish. b)Insertion - The noise present in the upper zone would be confused with *bind*i. Sometimes an upper zone vowel would be broken into smaller components, which would generate extra *bindi* characters. The above statistics are obtained without the application of the post processor. The recognition accuracy of the OCR without post processing was 94.35%, which was increased to 97.34% on applying the post processor to the recognized text.

Some other observations which were made during the experiments are :

a. The upper zone characters, which account for 28.6% of character occurrences have recognition accuracy of 91.19% which is improved to 95.14% by the post processor. The upper zone vowels, (ੇ and ੋ) , because of the similarity in shapes and large possible shape combinations were greatly confused by the recognizer and the confusion was partially resolved by the post processor. The other source of error in the upper zone is the character *bindi*, which has already been discussed in one of the previous sections.
b. The recognizer performed very well on the characters in the middle zone, which is the busiest zone. There are 42 characters in the middle zone and their combined frequency of occurrence is 68.2%. A majority of the errors in this zone were made in the recognition of the visually similar character pairs (ਥ and ਧ) and (ਚ and ਰ) and the characters with

dots at their feet. The recognition rate of middle characters is 96.71%, which is further improved to 98.38% by the post processor.

c. The lower zone, in which 2 vowels and 3 half characters reside, accounts for approximately 3.2% of total character population, proves to be the most difficult zone to remove errors. Some of the causes of the poor performance of the recognizer are the similarity in shapes of the characters, small size of the characters and merging of the lower zone characters with the middle zone characters. The recognition rate of the lower zone characters has been observed to be 79.48%, which has been improved to 87.87% by the post processor.

A sample text image page is shown in Fig. 12. Each text row in the image has been taken from different books and laser print outs printed in different fonts and sizes. Touching characters in zones as discussed in section 5 can be found in the text image. For example, lower zone characters touching middle zone characters are present in second word of third line, touching characters in upper zone are present in second and third words in fifth row and lower zone touching characters are present in last word of second last row. The sixth and seventh rows are horizontally overlapping. The output after thinning, line segmentation, classification and post processing stages is depicted in Fig. 13.

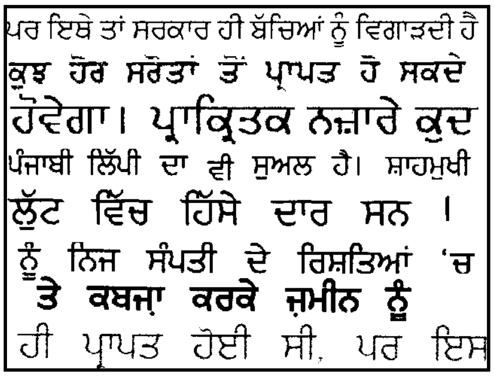
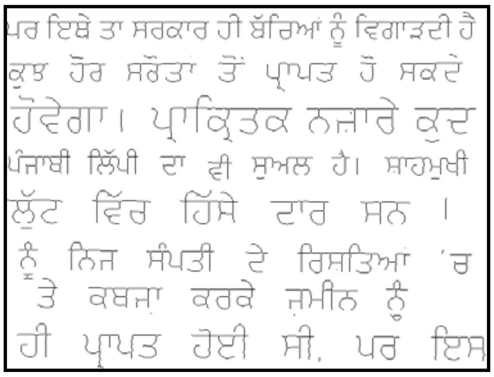

ਪਰ ਇਥੇ ਤਾਂ ਸਰਕਾਰ ਹੀ ਬੱਚਿਆਂ ਨੂੰ ਵਿਗਾੜਦੀ ਹੈ
ਕੁਝ ਹੋਰ ਸਰੋਤਾਂ ਤੋਂ ਪ੍ਰਾਪਤ ਹੋ ਸਕਦੇ
ਹੋਵੇਗਾ। ਪ੍ਰਾਕ੍ਰਿਤਕ ਨਜ਼ਾਰੇ ਕੁਦ
ਪੰਜਾਬੀ ਲਿੱਪੀ ਦਾ ਵੀ ਸੁਆਲ ਹੈ। ਸ਼ਾਹਮੁਖੀ
ਲੁੱਟ ਵਿੱਚ ਹਿੱਸੇ ਦਾਰ ਸਨ ।
ਨੂੰ ਨਿਜ ਸੰਪਤੀ ਦੇ ਰਿਸ਼ਤਿਆਂ ‘ਚ
ਤੇ ਕਬਜ਼ਾ ਕਰਕੇ ਜ਼ਮੀਨ ਨੂੰ
ਹੀ ਪ੍ਰਾਪਤ ਹੋਈ ਸੀ, ਪਰ ਇਸ

Fig. 12 : A sample text image


ਪਰ ਇਥੇ ਤਾਂ ਸਰਕਾਰ ਹੀ ਬੱਚਿਆਂ ਨੂੰ ਵਿਗਾੜਦੀ ਹੈ
ਕੁਝ ਹੋਰ ਸਰੋਤਾਂ ਤੋਂ ਪ੍ਰਾਪਤ ਹੋ ਸਕਦੇ
ਹੋਵੇਗਾ। ਪ੍ਰਾਕ੍ਰਿਤਕ ਨਜ਼ਾਰੇ ਕੁਦ
ਪੰਜਾਬੀ ਲਿੱਪੀ ਦਾ ਵੀ ਸੁਆਲ ਹੈ। ਸ਼ਾਹਮੁਖੀ
ਲੁੱਟ ਵਿੱਚ ਹਿੱਸੇ ਦਾਰ ਸਨ ।
ਨੂੰ ਨਿਜ ਸੰਪਤੀ ਦੇ ਰਿਸ਼ਤਿਆਂ ‘ਚ
ਤੇ ਕਬਜ਼ਾ ਕਰਕੇ ਜ਼ਮੀਨ ਨੂੰ
ਹੀ ਪ੍ਰਾਪਤ ਹੋਈ ਸੀ, ਪਰ ਇਸ

a)

ਪਰ ਇਥੇ ਤਾ ਸਰਕਾਰ ਹੀ ਬੱਚਿਆਂ ਨੂੰ ਵਿਗਾੜਦੀ ਹੈ   ਪਰ ਇਥੇ ਤਾ ਸਰਕਾਰ ਹੀ ਬੱਚਿਆਂ ਨੂੰ ਵਿਗਾੜਦੀ ਹੈ

ਕੁਝ ਹੇਰ ਸਰੋਤਾਂ ਤੋ ਪ੍ਰਾਪਤ ਹੋਂ ਸਕਦੋ
ਹੋਵੇਗਾ । ਪ੍ਰਾਕ੍ਰਿਤਕ ਨਜ਼ਾਰੋ ਕੁਦ
ਪੰਜਾਥੀ ਲਿੱਪੀ ਦਾ ਵੀ ਸੁਅਲ ਹੈ । ਸਾਹਮੁਖੀ
ਲੁੱਟ ਵਿੱਚ ਹਿੈੱਸੇ ਦਾਰ ਸਨ ।
ਨੂੰ ਨਿਜ ਸੰਪਤੀ ਦੇ ਰਿਸਤਿਆਂ ' ਚ
ਤੇ ਕਥਜਾ ਕਰਕੇ ਜ਼ਮੀਨ ਨੂੰ
ਹੀ ਪ੍ਰਾਪਤ ਹੇਈ ਸੀ , ਪਰ ਇਸ

b)

ਕੁਝ ਹੋਰ ਸਰੋਤਾ ਤੇ ਪ੍ਰਾਪਤ ਹੋ ਸਕਦੇ
ਹੋਵੇਗਾ । ਪ੍ਰਾਕ੍ਰਿਤਕ ਨਜ਼ਾਰੇ ਕੁਦ
ਪੰਜਾਬੀ ਲਿੱਪੀ ਦਾ ਵੀ ਸੁਅਲ ਹੈ । ਸਾਹਮੁਖੀ
ਲੂੱਟ ਵਿੱਚ ਹਿੱਸੇ ਦਾਰ ਸਨ ।
ਨੂੰ ਨਿਜ ਸੰਪਤੀ ਦੇ ਰਿਸ਼ਤਿਆਂ ' ਚ
ਤੇ ਕਬਜਾ ਕਰਕੇ ਜ਼ਮੀਨ ਨੂੰ
ਹੀ ਪ੍ਰਾਪਤ ਹੋਈ ਸੀ , ਪਰ ਇਸ

c)

Fig. 13 : Different stages in recognition of sample image of Fig. 12 a) Image after thinning b) Output of the classification process. Wrongly recognized characters are rendered in red colour c) Output after post processing

9. Conclusion

This is the first time that a complete multi-font and multi-size OCR system for Gurmukhi script has been developed. It has been tested on good quality images from books and laser print outs and has recognition accuracy of more than 97%. We are now working for the testing and for the improvement of the performance of the OCR on newspapers and low quality text.

References

1. V. K. Govindan, A. P. Shivaprasad, "Character recognition-A review", *Pattern Recognition*, Vol. 23, 1990, pp. 671-683.
2. S. N. S. Rajasekaran, B. L. Deekshatulu, "Recognition of printed Telugu characters", *Computer Graphics and Image Processing*, Vol. 6, 1977, pp. 335-360.
3. G. Siromoney, R. Chandrasekaran, M. Chandrasekaran, "Machine recognition of printed Tamil characters", *Pattern Recognition*, Vol. 10, 1978, pp. 243-247.
4. R. M. K. Sinha, H. N. Mahabala, "Machine recognition of Devanagari script", *IEEE Trans on Systems, Man and Cybernetics*, Vol. 9, 1979, pp. 435-449.
5. B. B. Chaudhuri, U. Pal, "A complete printed Bangla OCR system", *Pattern Recognition*, Vol. 31, 1998, pp. 531-549.
6. V. Bansal, "Integrating knowledge sources in Devanagri text recognition", *Ph.D. thesis*. IIT Kanpur, 1999.
7. G S Lehal and Chandan Singh, "Text segmentation of machine printed Gurmukhi script", *Document Recognition and Retrieval VIII*, Paul B. Kantor, Daniel P. Lopresti, Jiangying Zhou, Editors, Proceedings SPIE, USA, Vol. 4307, 2001, pp. 223-231.
8. Ajay Goyal, G S Lehal and S S Deol, "Segmentation of Machine Printed Gurmukhi Script", *Proceedings 9th International Graphonomics Society Conference*, Singapore, 1999, pp. 293-297.
9. G S Lehal and Chandan Singh, "Feature extraction and classification for OCR of Gurmukhi script", *Vivek*, Vol. 12, No. 2, 1999, pp. 2-12.

10. G. S. Lehal, C. Singh: A shape based post processor for Gurmukhi OCR. *Proceedings 6th International Conference on Document Analysis and Recognition*, Seattle, USA, 2001 1105-1109.
11. G S Lehal and Renu Dhir, “A Range Free Skew Detection Technique for Digitized Gurmukhi Script Documents”, *Proceedings 5th International Conference of Document Analysis and Recognition,* Bangalore, 1999, pp. 147-152.
12. W. H. Abdulla, A. O. M. Saleh and A. H. Morad, “A preprocessing algorithm for handwritten character recognition”, *Pattern Recognition Letters,* Vol. 7, 1988, pp. 13-18.