Optical Character Recognition of Gurmukhi Script using Multiple Classifiers

Gurpreet Singh lehal
Punjabi University, Patiala
India.
+91-98154-73767
gslehal@gmail.com

ABSTRACT
In this paper, we present a robust and font independent Gurmukhi OCR system, which performs reasonably well on old documents as well. The OCR is based on four classifiers operating in serial and parallel mode. For combining the results of the classifiers operating in parallel mode, a corpus based weighted voting method is used. Combining multiple classifiers in such a way, that their individual weaknesses are compensated while their individual strengths are preserved, results in significantly better performance than what can be achieved with a single classifier. The problem of broken characters, which frequently appear in old documents, has also been tackled using a structural feature based algorithm.

Keywords
Gurmukhi OCR, multiple classifiers, broken characters

1. INTRODUCTION
The idea of combining classifiers for increasing the recognition accuracy has been widely used in recent pattern recognition applications[1-5]. It has been experimentally shown that combination of several complementary classifiers leads to classifiers with improved performance. There are a number of classification algorithms available developed from different theories and methodologies. Usually, for a specific application problem, each of these classifiers could reach a certain degree of success, but generally none of them is totally perfect for practical applications. Similarly, for a specific recognition problem, there are often numerous types of features which could be used to represent and recognize patterns. These features are also represented in very diversified forms and it is rather hard to lump them together for one single classifier to make a decision. As a result, multiple classifiers are needed to deal with the different features [6]. It also results in a general problem how to combine those classifiers with different features to yield the improved performance.

In this paper, we present a multiple classifier based Gurmukhi OCR based on four classifiers operating in serial and parallel mode. For combining the results of the classifiers operating in parallel mode, a corpus based weighted voting method is used. We had earlier developed a Gurmukhi OCR, with recognition accuracy of around 97%, which used two classifiers in serial mode [7]. Though the OCR performed very well on good quality text, its accuracy went down for bad quality text images. To increase the accuracy of the OCR, particularly for poor quality documents, we added two more classifiers in parallel to the existing one. The experiments, which are discussed in following sections, have given very encouraging results. Another problem, we faced for recognition of old documents was the presence of broken characters. Many papers have appeared in literature for handling this problem[8-12]. Most of the techniques suggested in literature use character image restoration techniques simultaneously with the recognition modules. In this paper we present a structural feature based algorithm to detect and join broken word and character components in Gurmukhi text, which works independently of recognition module.

2. CHARACTERISTICS OF GURMUKHI SCRIPT
Gurmukhi script is used primarily for the Punjabi language, which is the world's 12th most widely spoken language. The structural properties of the Gurmukhi script, from OCR point of view are:

- 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 (੍ਹ ੍ਵ ੍ਰ), which lie at the feet of consonants. The complete Gurmukhi character set is shown in Fig 1.

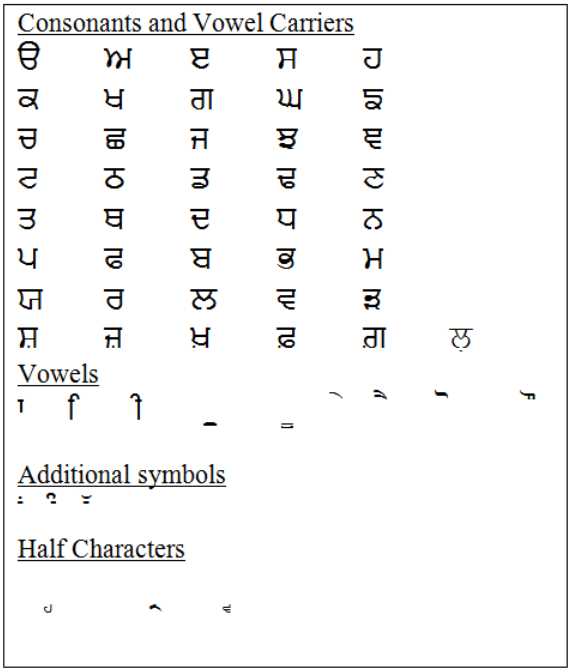


Figure 1. Gurmukhi Character Set

- A majority of the characters have a horizontal line at the upper part. The characters of words are connected mostly by this line called head line and usually there is no vertical inter-character gap in the letters of a word. The words are, however, separated with blank spaces.
- A word in Gurmukhi script can be partitioned into three horizontal zones (Fig 1). 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 lower zone represents the area below middle zone where some vowels and certain half characters lie in the feet of consonants.
- All the characters in the middle zone, touch the headline at least once.
- The vertical line if present in a character, is mostly present on the right most end of the character.
- The consonant height is usually greater than its width. The exceptions are ਅ and ਘ

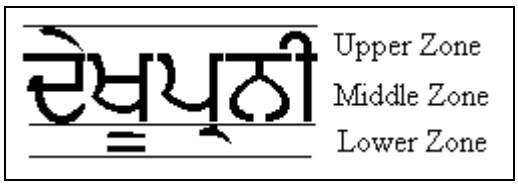


Figure 2. Three zones of a word in Gurmukhi script

3. SYSTEM ARCHITECTURE

In our present work, we have used four classifiers for recognition of the Gurmukhi text image. The first two classifiers operate in serial mode. The result of these classifiers is next combined in parallel mode with the other two classifiers using a rule based voting method. The system architecture is shown in Fig. 3.

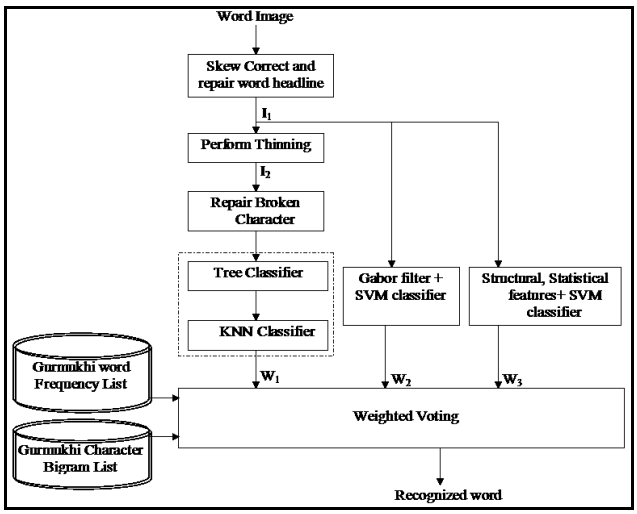


Figure3. Multi Classifier Gurmukhi OCR System Architecture

The word images are extracted from the text image and after skew correction and cleaning are sent to the recognizers for character segmentation, feature extraction and classification. It was found that frequently in old documents, the word images had broken headlines and characters, which resulted in low recognition accuracy. To increase the recognition accuracy it became necessary to repair these broken headlines and characters.

In our present work, we have developed a structural feature based algorithm to detect and join broken word and character components in Gurmukhi text. We tackle the breakage in words at two levels: breakage in headline and breakage in characters.

Figure 4. Sample word images with broken headlines

4. HANDLING BROKEN HEADLINES IN GURMUKHI WORD IMAGES

As discussed above, in a Gurmukhi word the middle zone characters are glued along the headline. The character segmentor expects the middle zone characters to be touching the headline, with no vertical inter-character gap. But in type written and old text, as well as locally skewed word images, the headline is frequently broken and the characters are not aligned along the y-axis as a result the word image is split into a group of single or multiple characters (Fig. 4). This creates problem for the character segmentor as it becomes difficult to identify the position of the headline and the components lying above and below it. Thus it becomes necessary to join the broken headlines and align the headline of the characters of the word by displacing the character images along the y-axis. To solve the problem, first the horizontal row in the word image corresponding to the word headline (*WHL*) in the upper half of the word image is determined, based on the following four factors:

4.1 Position of the headline in the sentence. This is determined as the horizontal row in the sentence image with having maximum number of black pixels.

4.2 Position of the headline in the previous two words. It may happen that there is some local skew in the word images due to warping or tilt at end of the sentences, as a result the word headline may not be aligned with the sentence headline.

4.3 Row with maximum horizontal span of black pixels in the word image.

4.4 Row having maximum number of black pixel count in the word image.

Once *WHL* is determined, the word image is decomposed into connected components (*CCs*) separated by vertical white space. Each of these *CC* could represent a group of one or more characters.

For each of the Component, determine the row corresponding to the position of component headline (*CHL*) satisfying one or both of the following criteria in the following order:

- Row with maximum horizontal span of black pixels.
- Row having maximum number of black pixel count.

- Row should be at most *d* pixels apart from *WHL* where *d* = 0.2*height of *CC*

Let *delta* = *WHL* – *CHL*
Move all the pixels of *CC* by *delta* pixels along y axis.
Lastly, join all the *CCs* by drawing a horizontal line between the *CCs* along *WHL*.
Fig. 5 displays some of the sample word images with broken headlines, which were joined back by applying the above algorithm.

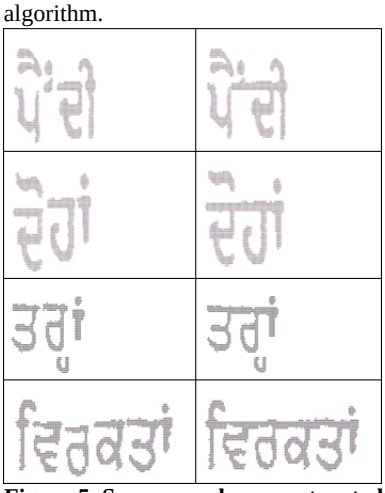


Figure 5. Some sample reconstructed broken word headlines

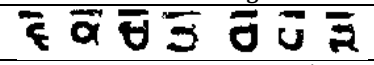
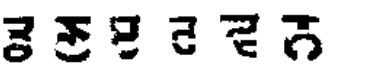
5. BROKEN CHARACTERS IN GURMUKHI

For the first two classifiers, which operate in serial mode, the image is thinned (reduced to single pixel width) before being sent for recognition. These classifiers use structural features and are thus highly sensitive to cuts and joints. Thus it was necessary to repair the broken characters in such skeletonized images. An in-depth analysis was made of the common broken characters in Gurmukhi. For this purpose about 2500 words containing broken characters scanned from old books were collected. The broken characters can be broadly categorized as:

- Characters broken from the headline
- Characters split vertically into non-overlapping bounding boxes
- Characters split horizontally into non-overlapping bounding boxes
- Character split into two or more parts with overlapping bounding boxes.

It is to be noted that we ignore the headline, while looking for overlapping regions.

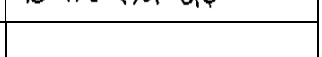
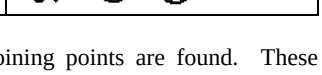
Table 1. Categories of broken characters in Gurmukhi

Cat.	Image
1	
2	
3	
4	

As already discussed above, a Gurmukhi word can be partitioned into three zones. We found that the majority of broken character segments are present in the middle zone. It is also to be noted that there are six multiple component characters in the middle zone and care has to be taken that we do not join those components. Also many times, the characters in the lower zone are very closely placed near the middle zone characters and they should not treated as broken components of middle zone characters and joined with them. The broken components have to be joined with the appropriate components to form the character. Decision also has to be taken which connected component pairs have to be joined and which were to be ignored. All this calls for a detailed study of the structure of Gurmukhi characters and words, while designing the algorithm.

First the position of the headline in the word image is noted and the headline is then rubbed off. The word image is then decomposed into connected components (CCs) and the relevant information about the CCs is extracted and stored. The closely lying CCs are determined. It is observed that in many cases, genuinely separated CCs, are lying very close to each other and care has to be taken that they are not joined. We have categorized such closely lying CCs, which should not be joined in Table 2.

Table 2. Closely lying Gurmukhi Connected Components to be ignored for joining

Category	CC Images with headlines removed
1. CCs of middle zone characters containing dot like symbol at feet of characters called *nukta*	
2. Closely lying CCs of the characters in middle and lower zone	
3. Vertically closely lying or overlapping CCs	
4. CCs which are vertically very close and one of the CC is a vertical line	

For the joinable pairs, their joining points are found. These joining points could be, depending on the overlapping category, endpoints, bend points, joints or boundary points in the CC.
We define the endpoint, joint, bend point and boundary point as follows:

- **End Point**

A black pixel not lying on the y-axis corresponding to the headline and with only one black pixel in its 3 x 3 neighbourhood.

- **Joint**

A black pixel not lying on the y-axis corresponding to the headline and having three or more black pixels in its 3 x 3 neighbourhood.

- **Bend Point**

A white pixel where two or more lines meet at 90 or 45 degrees.

- **Boundary Point**

A black pixel lying on one of the boundary of the CC.

These points are extracted from the CCs and the point pairs, where the first point is from one CC and second point from the other CC, lying within the threshold value are collected. If no such pair is found then the CCs are not joined. The decision to

join the CCs is kept pending, till all the CC pairs have been processed. If any of the joining points have some common points, then only the nearest pair is retained. The points are then joined by drawing lines between them. If there remain some CCs, which are not touching the headline, we use the structural property of Gurmukhi script that all the characters in the middle zone touch the headline at least once and increase the threshold value to test if they can be joined with any other CC or headline. The complete algorithm is described in the next section.

6. Algorithm for Repairing Skeletonized Gurmukhi Characters

The algorithm to repair the skeletonized Gurmukhi characters is as follows:

1. Skeletonize the word image.
2. Determine the position of the headline in the word image and rub off the headline.
3. Decompose the word image into connected components. Create the list, *CList*, of all the connected components (CCs) which lie horizontally below the position of headline in the middle zone. For each of the CC, store the information about the global position of its bounding box, number and position of end points, joints, bend points and boundary points in the CC and other such relevant information.
4. Sort the CCs along the x-axis. Find the heights of the CCs touching the headline. Set *threshold* = (height of the tallest CC) / 4.
5. Find all pair of CCs, whose bounding boxes are at most *threshold* distance apart. Set their *overlap type* as
   - 0 if the bounding boxes overlap horizontally and vertically
   - 1 if the bounding boxes share some common points along x-axis, but not along y-axis.
   - 2 if the bounding boxes share some common points along y-axis, but not along x-axis.
   - 3 if the bounding boxes do not share any common points in x or y axis.
6. If for any pair of CCs, CC1 and CC2, if CC1 lies in the lower quarter of the word image along y-axis, check that:
   - The minimum value of bounding box of CC1 along y-axis should not be greater than the median height of the CCs touching the headline.
   - The area of CC1 should be at least 2**threshold* pixels.
   - If the *overlap type* is 0, then CC1 should not lie in the middle or left of CC2.
7. If any of the above condition is false, remove the connected component pair from the list. This is to avoid joining the nukta signs and the characters lying in lower zone with the middle zone characters (Category 1 and 2 of Table2).
8. If overlap type is 0, set *thresh* = 1.25**threshold*.
9. Let S1 = Set of all endpoints, bend points and joints in CC1.
10. Let S2 = Set of all endpoints, bend points and joints in CC2.
11. Find C= Set of all pairs of points from S1 and S2 which lie within *thresh* distance.
12. If for any pair, their bounding boxes overlap, retain the smaller distance pair and delete the other pair to avoid creation of cycles.
13. If overlap type is 1 or 3, set *thresh* = *threshold*. Assume CC1 is nearer to headline.
14. Let S1 = Set of all endpoints, bendpoints, joints and boundary points lying along the lower horizontal boundary of the bounding box of CC1.
15. Let S2 = Set of all endpoints, bend points, joints and boundary points lying along the upper horizontal boundary of the bounding box of CC2.
16. Find C= Set of all pairs of points from S1 and S2 which lie within *thresh* distance.
17. If overlap type is 2, set *thresh* = *threshold*. Assume CC1 occurs before CC2 in the sorting order along x-axis. If CC1 is a vertical line or the width to height ratio of the bounding box formed by joining CC1 and CC2 is greater than 1.25 then ignore this pair for merging. This is to take care that the closely lying characters are not merged together (Category 3 and 4 Table2).
18. Let S1 = Set of all endpoints, bendpoints, joints and boundary points lying along the right vertical boundary of the bounding box.
19. Let S2 = Set of all endpoints, bend points and joints and boundary points lying along the left vertical boundary of the bounding box of CC2.
20. If CC2 is a vertical line then exclude the joints and boundary points from S1. This is to avoid joining CCs similar to first and third pairs of CCs of category 4 in Table 2, as their nearest joining points lying within the threshold are on boundary or on joints.
21. Find C= Set of all pairs of points from S1 and S2 which lie within *thresh*. If both the points in a pair are boundary points, remove that pair.
22. Examine all the candidate pairs in set C. If any pair has a common co-ordinate point, retain the one having smaller distance.
23. At the end join all the candidate pairs in set C by drawing a straight line between them. Update *CList*, the list of Connected Components, by replacing the CCs which have been joined with the new merged CC.
24. Let *SList* be the sublist of *CList* containing connected components not touching the headline. If *SList* is not empty, then add the skeletonized image of headline to *CList*.
25. If *SList* empty then Stop, else
26. Set *thresh* = 1.5 * *threshold*. Repeat the steps 5 to 11 and then stop. Instead of taking both pairs of CCs from *Clist*, one element is from *CList* and second from *SList*. This is to ensure the Gurmukhi Script property that all the characters in the middle zone, touch the headline at least once. So if we find a CC not touching the headline, then it is a candidate for joining with either the headline or some other CC which is touching the headline.

The algorithm was tested on a set of 2500 words containing broken characters and in 82.3% of cases, the broken components were correctly joined together to form a recognizable unit, while in 4.9% of cases, the components were wrongly joined and in

12.8% of cases the broken components were not joined with any other component. Some sample images are shown in Fig. 6. The first image in each row is the thinned binary image of the word. The second image is the image obtained after joining the broken components and smoothening the headline.

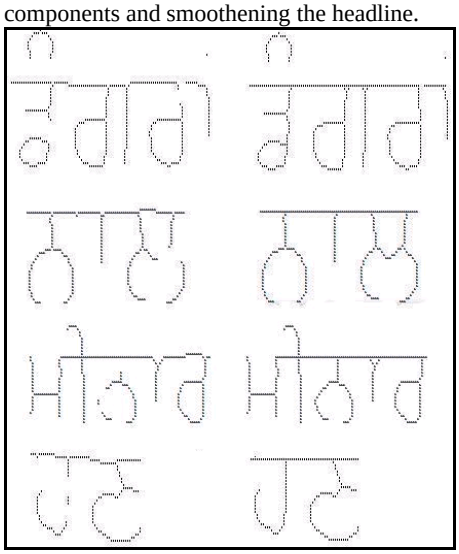


Figure 6. Sample broken character images repaired

7. CLASSIFIERS USED IN GURMUKHI OCR

As already mentioned, we have used four classifiers in our Gurmukhi OCR system. For the first two classifiers, which operate in serial mode, the image is skeletonized before being sent for recognition. As an example, we have in Fig. 7(a), a word image for recognition. The image after skew correction and headline repair is sent to the C2 and C3 classifiers is in Fig. 7 (b), while the same image after skeletonization and headline smoothening (Fig 7c) is sent is to C1 classifier. The classifier C1 is a combination of two classifiers (Binary Tree Classifier and KNN Classifiers) , which operate in serial mode. The details are available in [7].

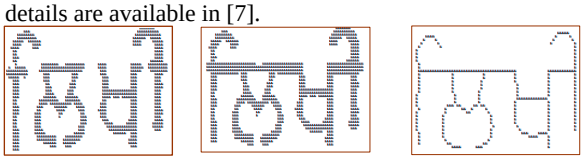


Figure7.
a) Original Image b) Repaired Image c) Thinned Image

The binary tree classifier, operates on middle zone characters and its purpose is to classify the patterns into classes of similar looking characters. The purpose is to precisely divide the set of characters lying in middle zone into smaller subsets which can be easily managed by the kNN classifier. The kNN Classifier next classifies the patterns to the final class. The binary tree classifier uses the following Boolean valued structural features:

- Number of junctions with the headline
- Presence of sidebar
- Presence of a loop
- Presence of a loop along the headline

The KNN classifier operates on the following structural features:

- Number of endpoints and their location
- Number of junctions and their location
- Horizontal Projection Count
- Left and Right Projection profiles
- Left and Right Profile depth
- Left and Right Profile Direction Code
- Aspect Ratio
- Distribution of black pixels about the horizontal mid line

Classifier C2, which is an SVM classifier, uses Gabor filters as features and the feature vector size is **189**.

Classifier C3, is also SVM classifier and it operates on the following structural and statistical features:
The number in the brackets represents the feature vector size.

- Left, right, top and bottom profile direction codes (**12**)
- Directional Distance Distribution (**144**)
- Black to White Transition (**100**)
- Zoning (**49**)

8. CLASSIFIER COMBINATION

There have been extensive studies on the combination of results multiple classifiers. There are two architectural methodologies for classifier combination. In multi-expert methods, the classifiers work in parallel. All the classifiers are trained on all patterns and given a pattern, they all give their decisions and a separate combiner computes the final decision. Multistage methods use a serial approach where the next classifier is trained/consulted only for patterns rejected by the previous classifiers.

For multi expert systems, voting is the simplest combination method[13-14]. Voting considers only the best class output by each classifier and regards the class which appears most often in the output of the classifiers as output of the combined classifier. For classifiers with different performances, weighted voting is more appropriate, which assigns a weight wt_i to each classifier *Ci* and regard the class which has the highest sum of weights of classifiers outputting this class as the result of the combined classifier.

We have used a rule based weighted voting classification scheme, which uses Gurmukhi corpus for decision making. As shown in Fig. 3, the output of the three classifiers (C1, C2 and C3) are recognized words w1, w2 and w3 respectively. The words are then combined to give the final recognized word as discussed in the following algorithm.

From experiments it was found that the classifier C3 had highest recognition accuracy followed by C2 and C1 classifiers in that order. Thus highest priority is given to word w3 followed by w2 and w1. The algorithm makes use of Punjabi word frequency list generated from a 10 million word Punjabi corpus and a character bigram list generated from the same corpus. The word frequency list guides in word selection while the character bigram list is used to detect illegal character combinations. The algorithm works as follows:

First we search for the three words in the word frequency list. If any two words are same and found in the list then they are retained as final word, else if only one of the word is found or if more than one word found but they are not same then the words are selected according the priority assigned to them. It was also observed that due to thinning and small size of the characters in upper zone, some of the similar characters in upper zone were confused by **C1** but the same characters are usually recognized properly by **C2** and **C3**. Thus if for any upper zone character, C2

and C3 agree on same character, then their decision is given higher weight age as compared to C1.If none of the word is found in the frequency list, then we generate combinations of words from the three words and select the word with highest frequency.

The algorithm is as follow:

Input

1) Recognized words from C1, C2 and C3 classifiers. (w1, w2 and w3 respectively)
2) Set of Unigrams extracted from a 10 million word Punjabi corpus. (UG)
3) Set of valid bigram character combinations

Output

Final recognized word

Algorithm

- if (w1=w2=w3) return w1
- If w1, w2 and w3 ϵ UG then if
  - w1 = w2 return w1 (Rule 1.1)
  - w1 = w3 return w1 (Rule 1.2)
  - w2 = w3 return w2 (Rule 1.3)
- If w3 ϵ UG return w3 (Rule 2)
- If w2 ϵ UG return w2 (Rule 3)
- If w1 ϵ UG then
  - Let uw1[], uw2[] and uw3[] be the set of upper characters in w1, w2 and w3 respectively.
  - If (w2 ϵ UG) or (w3 ϵ UG) then
    if for any i (uw1[i] <> uw2[i]) and (uw2[i]==uw3[i]) then uw1[i] = uw2[i]
  - return w1 ((Rule 4)
- Else if // when none of w1, w2 or w3 ϵ UG
  - w1 = w2 return w1 (Rule 5.1)
  - w1 = w3 return w1 (Rule 5.2)
  - w2 = w3 return w2 (Rule 5.3)
- Else Create a list of all valid combination of words formed by taking individual characters from each word in same zonal position as follow:
- Create a tree with the root node containing empty string and at each level the children nodes are added and the words stored in them are formed by concatenating the word stored in parent node with the non-unique characters in same corresponding zonal position in the three words. If any of the new word formed contains an invalid bigram character, then that branch is cut off and not further expanded. At the end the words in the leaf nodes are considered and the word with highest frequency of occurrence is returned as final word. (Rule 6)
- In case no valid word found in above step, then count the invalid bigram character combinations in w1, w2 and w3. Return the one with least count of invalid combinations. In case of tie, return the word with higher priority. (Rule 7)

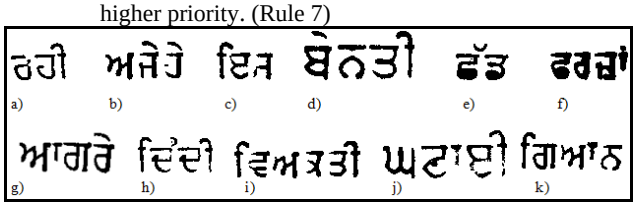


Figure 8. Some sample images

Table 3. Classifier combination output

Image	C1	C2	C3	Final	Rule
A	ਠਹੀ	*ਰਹੀ*	ਠਹੀ	ਰਹੀ	3
B	*ਅਜੇਹੇ*	ਅਜੇਤੇ	ਅਜ਼ੇਹੇ	ਅਜੇਹੇ	4
C	ਇਜ	*ਇਸ*	ਇਰ	ਇਸ	3
D	*ਬੇਨਤੀ*	*ਬੇਨਤੀ*	ਥੇਨਤੀ	ਬੇਨਤੀ	1.1
E	*ਦੱਤ*	*ਛੱਡ*	*ਛੱਡ*	ਛੱਡ	1.3
F	ਟਰਚਾਂ	ਛਝਭਾਂ	*ਫਰਜ਼ਾਂ*	ਫਰਜ਼ਾਂ	2
G	ਆਗਰੇ	ਆਗਥੇ	ਆਗਰੇ	ਆਗਰੇ	5.2
H	ਦਿੱਂਦੀ	ਦਿੇਦੀ	ਟਿੰਟੀ	ਦਿੰਦੀ	6
I	ਵਿਅਤਤੀ	ਵਿਅਝਤੀ	*ਵਿਅਕਤੀ*	ਵਿਅਕਤੀ	2
J	*ਘਟਾਈ*	ਘਟਾਟੀ	ਘਟਾਟੀ	ਘਟਾਈ	4
K	ਜਿਆਨ	ਝਿਆਨ	ਗਿਆਠ	ਗਿਆਨ	6

For illustration purpose, we have some sample images taken from Gurmukhi text. The quality of some of the images is not good, including some broken and heavy printed images. Table 3, depicts the words recognized by the three classifiers as well the final word after combining the classifier results. The rule used for combination is also displayed in the table. The words found in the frequency list are italicized. It can be seen in Table 3, that individually the classifiers are not able to correctly recognize the images, but on combining the results the final word is correctly classified. We will like to make special mention of images. First we consider the word image in Fig 8h. It is recognized by the three classifiers as, w1= ਦਿੱਂਦੀ, w2 = ਟਿੰਟੀ and w3 = ਦਿੇਦੀ. As none of w1, w2 and w3 ϵ UG and no two words are same, so we find the combinations of all the valid words that can be formed from corresponding characters in same zonal positions in w1, w2 and w3 as shown in Fig. 9. The leaf nodes have four words and as the word ਦਿੰਦੀ has the highest frequency of occurrence, so it is finally selected. Similarly, the image in Fig. 8k, is wrongly recognized by the three classifiers. But when the characters at similar zonal positions are combined from the three words and the resulting words are checked in the frequency list, we get the correct final word. In another example of image in Fig. 8j, we see how simple majority voting fails, while the lexicon lookup helps in identifying the correct word. The image is wrongly recognized by classifiers C2 and C3. Even though w2 and w3 are same but since they are not present in the lexicon and w1 is found in the lexicon, so w1 is selected as final word according to rule 4.

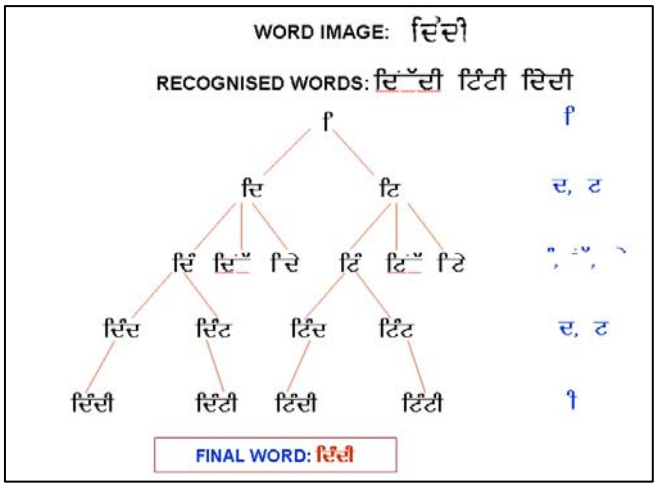


Figure 9. Combination tree for a sample image

It was found from experiments that the three classifiers, which operate in parallel, complement each other, very well. Particularly C1 complements C2 and C3. This is clear from the complementary table, discussed in next section.

It was observed that classifier C1, which operates on thinned images and uses structural features, is font and size independent and works very well for average or good quality text. But its limitations are: low recognition accuracy for upper zone characters which are usually small sized and after thinning resemble each other, and failure to correctly recognize heavy printed characters, characters with broken loops or extra loops or badly broken characters. On the other hand the other two classifiers, which use combination of statistical and structural features are font dependent but have better recognition accuracy for upper zone characters, as well as heavy printed and broken characters. Thus when the results of the recognizers are combined, in most of the cases the words wrongly recognized by individual classifiers were finally corrected.

9. EXPERIMENTAL RESULTS

We have tested the system on 31 pages taken from different sources containing 42650 characters and 10 different font types. The recognition accuracy of the classifiers is shown in Table 4. It can be noted that that C3 classifier gives best recognition accuracy of **96.51%.** On combining the outputs of the three classifiers, the accuracy increased to **98.18%,** an increase of **1.67%** over the best classifier.

Table 4 : Classifier Recognition Accuracy

Classifier	**Percentage Recognition Accuracy**
C1	94.05
C2	95.61
C3	96.51
Combined	**98.18**

A graph is also plotted in Fig.10, showing the recognition accuracy of all the classifiers on the 31 pages. The minimum recognition accuracy for C1, C2, C3 and combined classifiers is 88.17%, 89.27%, 90.41% and 93.60% respectively, while the maximum recognition accuracy for C1, C2, C3 and combined classifiers is 97.88%, 98.79%, 99.31% and 99.84% respectively. As discussed earlier, the classifier C1 had been used in the previous version of Gurmukhi OCR[7]. Thus we have a substantial gain of 4.13% in recognition accuracy over the present OCR using the combination of classifiers.

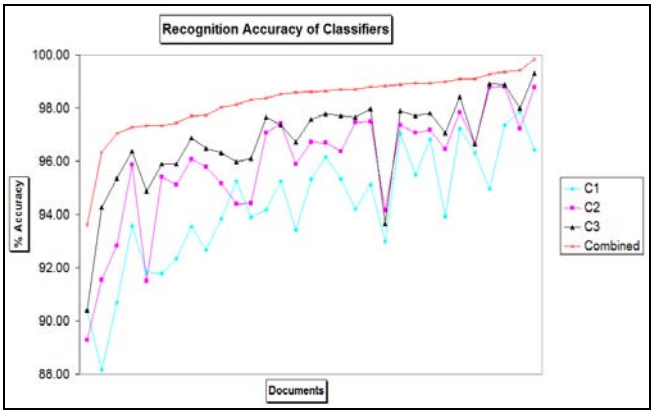


Figure 10. Recognition Accuracy of the Classifiers

We also analysed the combined performance of the classifiers at word level, since the basic input and output unit used for classifier combination is the word. Table 5 provides quick answers to questions such as : If all the three classifiers agree on the same word, then what is the recognition accuracy or if none of the classifier agrees on a common word, then what are the chances of correctly recognizing the word. The first column in Table displays the common words recognized by the classifiers. The second , third and fourth columns display the total count, correct word count and percentage of correctly recognized words respectively. The last count gives the percentage of occurrence of the words. Thus from the first row, we conclude that if all the three words are same, then in 99.57% cases, the word has been correctly recognized. In 79.74% of cases the three classifiers give the same output. We can also observe from the table that in 12.7% of cases classifiers C2 and C3, which operate on same unthinned image give the same output, though it is correct in only 72.32% of cases. On the other hand, if classifier C1 and C2 agree on same word then in 93.92% of cases the word has been correctly recognized. In only 0.81% of cases classifiers C1 and C2 give the same output and C3 gives different output. If all the three classifiers give different outputs then the chances of correct classification is only 59%.

Table 5 : Similar Word Analysis

Similar Words	**Count**	**Correct**	**Percentage Recognition Accuracy**	**Percentage Count**
w1, w2, w3	8790	8752	99.57	79.64
w1 and w2	89	72	80.89	0.81
w1 and w3	263	247	93.92	2.38
w2 and w3	1402	1014	72.32	12.70
None	493	291	59.02	4.47
Total	11037	10376	94.01	100

Next, we examine just how different the errors of the classifiers are. For this, we use the complementary error rate. Complementary rate of classifiers A and B, Comp(A,B), measures the percentage of time when classifier A is wrong and

classifier B is correct. In Table 6 we show the complementary rates between the different classifiers. For instance, when the C1 classifier is wrong, the C3 classifier is right 61.63% of the times, and when the C3 classifier is wrong, the C1 classifier is right 25.9% of the times.

The complementary rates are quite high, which is encouraging, since this sets the upper bound on how well we can do in combining the different classifiers. If all classifiers made the same errors, or if the errors that lower-accuracy classifiers made were merely a superset of higher accuracy classifier errors, then combination would be futile.

As already discussed before, classifiers C1 and C2 operate on different versions of same image, C1 operates on thinned images while C2 operates on unthinned images. On the other hand, C2 and C3 operate on same images. Thus it is interesting to see that even though overall C1 has the lowest recognition accuracy still in 25.9% of cases it correctly classifies the characters wrongly classified by C3, while C2 even though it has better recognition accuracy than C1, correctly recognizes 13.32% characters wrongly classified by C3.

As already discussed above, it was found that classifier C1 performs poorly on upper zone characters. This was borne out in table 7, where we analysed the performance of the classifiers on upper zone characters. As can be seen in 79.84% of cases classifier C3 correctly classified upper zone characters when C1 was wrong and only in 12.25% of cases C1 correctly classified the wrongly classified upper zone characters by C3. For middle and lower zone characters, C3 rightly classifies 51.14% of characters wrongly classified by C1, while C1 correctly classifies 34.11% of characters wrongly classified by C3.

Table 6: Complementary Table for all characters

Comp (Ci, Cj)	C1	C2	C3
C1	0	54.17	61.63
C2	33.43	0	34.26
C3	25.9	13.32	0

Table 7: Complementary Table for upper zone characters

Comp (Ci, Cj)	C1	C2	C3
C1	0	70.94	79.84
C2	19.25	0	32.76
C3	12.25	18.93	0

Table 8: Complementary Table for middle and lower zone characters

Comp (Ci, Cj)	C1	C2	C3
C1	0	44.63	51.14
C2	41.51	0	11.45
C3	34.11	34.78	0

For illustration purpose, we have a sample image in Fig. 11. The recognition results of the different recognizers are shown in Fig 12. Characters in red represent wrongly recognized characters.

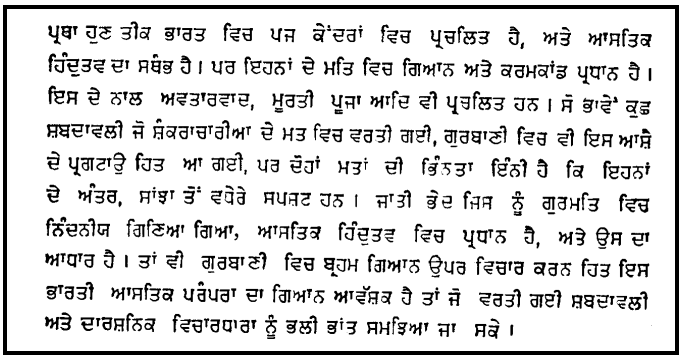


Figure11. A Sample Image

a) Output from Classifier C1 (Recognition accuracy 95.29%)

b) Output from classifier C2 (Recognition accuracy 94.47%)

c) Output for Classifier 3 (Recognition accuracy 97.13%)

ਪ੍ਰਥਾ ਹੁਣ ਤੀਕ ਭਾਰਤ ਵਿਚ ਪਜ ਕੇਂਦਰਾਂ ਵਿਚ ਪ੍ਰਚਲਿਤ ਹੈ, ਅਤੇ ਆਸਤਿਕ
ਹਿੰਦੁਤਵ ਦਾ ਸਥੰਭ ਹੈ। ਪਰ ਇਹਨਾਂ ਦੇ ਮਤਿ ਵਿਚ ਗਿਆਨ ਅਤੇ ਕਰਮਕਾਂਡ ਪ੍ਰਧਾਨ ਹੈ।
ਇਸ ਦੇ ਨਾਲ ਅਵਤਾਰਵਾਦ, ਮੂਰਤੀ ਪੂਜਾ ਆਦਿ ਵੀ ਪ੍ਰਚਲਿਤ ਹਨ। ਸੋ ਭਾਵੇਂ ਕੁਛ
ਸ਼ਬਦਾਵਲੀ ਜੋ ਸ਼ੰਕਰਾਚਾਰੀਆ ਦੇ ਮਤ ਵਿਚ ਵਰਤੀ ਗਈ, ਗੁਰਬਾਣੀ ਵਿਚ ਵੀ ਇਸ ਆਸ਼ੇ
ਦੇ ਪ੍ਰਗਟਾਉ ਹਿਤ ਆ ਗਈ, ਪਰ ਦੋਹਾਂ ਮਤਾਂ ਦੀ ਭਿੰਨਤਾ ਇੰਨੀ ਹੈ ਕਿ ਇਹਨਾਂ
ਦੇ ਅੰਤਰ, ਸਾਂਝਾਂ ਤੋ ਵਧੇਰੇ ਸਪਸ਼ਟ ਹਨ। ਜਾਤੀ ਭੇਦ ਜਿਸ ਨੂੰ ਗੁਰਮਤਿ ਵਿਚ
ਨਿੰਦਨੀਯ ਗਿਣਿਆ ਗਿਆ, ਆਸਤਿਕ ਹਿੰਦੁਤਵ ਵਿਚ ਪ੍ਰਧਾਨ ਹੈ, ਅਤੇ ਉਸ ਦਾ
ਆਧਾਰ ਹੈ। ਤਾ ਵੀ ਗੁਰਬਾਣੀ ਵਿਚ ਬ੍ਰਹਮ ਗਿਆਨ ਉਪਰ ਵਿਚਾਰ ਕਰਨ ਹਿਤ ਇਸ
ਭਾਰਤੀ ਆਸਤਿਕ ਪਰੰਪਰਾ ਦਾ ਗਿਆਨ ਆਵੱਸ਼ਕ ਹੈ ਤਾਂ ਜੋ ਵਰਤੀ ਗਈ ਸ਼ਬਦਾਵਲੀ
ਅਤੇ ਦਾਰਸ਼ਨਿਕ ਵਿਚਾਰਧਾਰਾ ਨੂੰ ਭਲੀ ਭਾਂਤ ਸਮਝਿਆ ਜਾ ਸਕੇ।

d) Output after combining results of all the recognizers (Recognition accuracy 99.59%)

Figure 12. Recognized text of sample image of Figure 11.

10. ACKNOWLEDGEMENT

This research work is sponsored by Ministry of Communications and Information Technology under the project : Development of Robust Document Analysis and Recognition System for Printed Indian Scripts**.**

11. REFERENCES

[1] Brill Eric and Jun Wu : Classifier Combination for Improved Lexical Disambiguation, Proceedings of the 17th international conference on Computational linguistics, vol.1, pp. 191-195. Montreal, Quebec, Canada (1998).

[2] Roli Fabio, Giacinto Giorgio, Vernazza Gianni : Methods for Designing Multiple Classifier Systems, Proceedings of the Second International Workshop on Multiple Classifier Systems, pp. 78 – 87. Springer-Verlag London, UK (2001).

[3] Ludmila IK., : A Theoretical Study on Six Classifier Fusion Strategies, IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 24, No. 2, pp. 281--286, (2002.).

[4] Kittler J., Hatef M., Duin RPW, Matas J. : On Combining Classifiers, IEEE Trans. On Pat. Analysis and Machine Intel., vol. 20, No.3, pp. 226—239 (1998).

[5] Prevost L., Michel-Sendis C., Moises A., Oudot L., Milgram M : Combining model-based and discriminative classifiers: application to handwritten character recognition, ICDAR'03 (2003).

[6] Ke Chen1, Lanwang, Huisheng Chi : Methods of Combining Multiple Classifiers with Different Features and their Applications to Text-Independent Speaker Identification: International Journal of Pattern Recognition and Artificial Intelligence, Vol. 11, No. 3, pp. 417-445 (1997).

[7] Lehal G. S., Singh Chandan, : A Complete Machine Printed Gurmukhi OCR System, Vivek, pp. 10--17, Vol. 16, No. 3. (2006).

[8] Benedicte Allier, Nadia Bali, Hubert Emptoz : Automatic accurate broken character restoration for patrimonial documents. IJDAR 8(4), pp 246--261 (2006)

[9] Billawala, N., Hart, P.E., Pearis, M. : Image continuation. In : Proceedings of the International Conference on Document Analysis and Recognition, pp. 53-57, Tsukuba, Japan (1993)

[10] Shi, Z., Govindaraju, V. : Character image enhancement by selective region-growing. Pattern Recognit. Lett. (17), pp. 523--527 (1996)

[11] Yu. D., Yan, H.: Reconstruction of broken handwritten digits based on structural morphological features. Pattern Recognit. (34), pp. 235--254 (2001).

[12] Whichello, A., Yan, H. : Linking broken character borders with variable sized masks to improve recognition. Pattern Recognition 29(8), pp. 1429--1435(1996)

[13] Bhattacharya U., Chaudhuri B. B. : A Majority Voting Scheme for Multi resolution Recognition of Hand printed Numerals, ICDAR'03, pp. 16--20, 3-6 (2003).

[14] Lam L., Suen, C. Y., : Application of Majority Voting to Pattern Recognition : An Analysis of its Behaviour and Performance, IEEE Trans. on System Man and Cyebrn-Part A : Systems and Humans, vol. 27, pp. 553 – 568 (1997).