OPTICAL CHARACTER RECOGNITION FOR PRINTED HINDI TEXT IN DEVNAGARI USING SOFT-COMPUTING TECHNIQUE

Divakar Yadav  
JIIT (Deemed Univ.), Noida  
India  
divakaryadav@rediffmail.com

Prof.A.K. Sharma  
YMCA, Faridabad  
India  
ashokkale2@rediffmail.com

Prof. J.P. Gupta  
JIIT (Deemed Univ.), Noida  
India  
jpgupta@jiit.ac.in

ABSTRACT
In this paper, we present an OCR for printed Hindi text in devnagari script. Text written in Devnagari script, there is no separation between the characters. Hindi is one of the most spoken language in India. About 300 million people speak Hindi in India. One of the important reasons for poor recognition rate in optical character recognition (OCR) system is the error in character segmentation.

Preprocessing task considered in this paper is conversion of gray scale images to binary images, image rectification, and segmentation of text into lines, words and basic symbols. Basic symbols are identified as the fundamental unit of segmentation in this paper which are recognized by neural classifier.

We have used three feature extraction techniques namely, histogram of projection based on mean distance, histogram of projection based on pixel value and Vertical Zero crossing. These feature extraction techniques are very much powerful to extract feature of even distorted characters.

A back-propagation neural network with two hidden layer is used to create a character recognition system. The system is trained and evaluated with printed text. A performance of approximately 90% correct recognition is achieved.

KEY WORDS
OCR, Pre-processing, segmentation, Feature Vector, Classification, Artificial Neural Network (ANN).

1. INTRODUCTION
1.1 Definition
Optical Character recognition (OCR) is a process of converting a printed document or scanned page into ASCII character that a computer can recognize [1]. The advantage being that the textual material can be edited. The document image itself can be either machine-printed or handwritten, or a combination of the two. Computer systems equipped with such an OCR system improve the speed of input operation, decrease some possible human errors and enable compact storage, fast retrieval and other file manipulations. The range of applications includes postal code recognition, automatic data entry into large administrative system, banking, automatic cartography and reading device for the visually handicapped when interfaced with a voice synthesizer.

1.2 Overview
For certain language scripts (e.g. Roman script) today, it is not difficult to develop an Optical Character Recognition (OCR) system that recognizes well-shaped and well-spaced characters with accuracy of 99% and above. However, it is still challenging to design a system that can maintain such high recognition accuracy, regardless of the quality of the input document and character font style variation.

In this paper, our concern is Devnagari script. It is the script for Hindi which is the official language of India. Many works on Devnagari script OCR have been reported [3, 4, 5, 6, 7]. However, few of these works have considered real life printed Hindi text in Devnagari consisting of character fusions and noisy environment.

Segmentation and classification are the two primary phases of a text recognition system. The segmentation process extracts recognition units from the text. The recognition unit is usually a character. The classification process computes certain features for each isolated character and each character is classified to a class that may be the true class (correct recognition), wrong class (substitution error), or an unknown class (rejection error).

Accuracy, flexibility and speed are the main features that characterize a good OCR system. Performance of most of the OCR systems, developed for Hindi, has been constrained by the dependence on the font, size and orientation. The recognition rate in these algorithms depends on the choice of features.

In this paper, first we have developed a KL divergence based classifier for recognizing Hindi text. Though its performance for handwritten character was quiet good but it failed in case of printed text. The reasons for the poor performance are analyzed and discussed later. Then we switched over to neural network based method for the same recognition task. Neural network based Hindi OCR, effectively reduces the image processing time while maintaining efficiency and versatility. Neural network approaches have been used for character recognition [2, 9], but a complete system that encompasses all the features of a practical OCR system is yet to be realized. The key factors involved in the
A horizontal line is drawn on the top of all characters of a word that is referred to as the header line or Shirorekha. It is convenient to visualize Devnagari words in terms of three strips: a core strip, a top strip and a bottom strip. The middle and upper zones are separated by the header line. Figure 1.2 shows the image of a sentence that contains three characters, one top modifier, and one lower modifier.

Fig. 1.2 Three Zones of Devnagari Script

1.4 Analysis of Previous works
During past 30 years, substantial research effort has been devoted to optical character recognition for the Devnagari script [3, 4, 5, 6, 7]. But till date no complete OCR for devnagari is available which works in noisy environment. One of the important reasons for poor recognition rate in OCR system is the error in character segmentation [3]. Existence of touching characters in the scanned documents is a major problem to design an effective character segmentation procedure.

The technique for character segmentation treats characters touching each other as a single character and leads to the failure in the character recognition phase. Faxed documents, photocopies, old books, newspapers, etc., contain a considerable number of touching characters. The module for automatic separation of touching characters is essential for successful OCR of such documents.

2. THE SEGMENTATION

Segmentation is one of the most important phases of OCR implementation. It directly affects the efficiency of any OCR. So a good segmentation technique can increase the performance of OCR. Basically in segmentation what we do is that we try to extract basic components of the script, which are certainly characters. This is needed because our classifier recognizes these characters only. In this research work, before applying segmentation algorithm on any text, we have done some pre-processing work and that is text digitization and skew correction.

2.1 Text Digitization and Preprocessing Techniques
In our system text digitization is done using a flatbed scanner that has a resolution between 100 and 600 dpi. The digitized images are usually in gray tone, and for a clear document, a simple histogram-based thresholding approach is sufficient for converting them to two-tone images. When a document page is fed to the scanner, it may be skewed by few degrees. The pages of thick book create more problems since the scanned image may both be distorted and skewed. There exist a wide variety of skew detection algorithms based on projection profile [8,10].
2.2 Line, Word and Character Segmentation
Once a text block as shown in Figure 2.1 is detected, the system automatically finds individual text lines, segments the words, and then separates the characters accurately.

Fig. 2.1 Text block of Devanagari.

2.2.1 Segmentation of Line
Text lines are detected by horizontal scanning. We scan horizontally from the top of the scanned document page and find the last row containing all white pixels, before a black pixel is found. Then we find the first row containing entire white pixel, just after the end of black pixels. We repeat this process on entire page to find out all lines. Figure 2.2 shows the output when this algorithm is applied on the text block given in Figure 2.1.

Fig. 2.2 Segmented line from the above text block

2.2.2 Segmentation of Words
After finding a particular line, we separate individual words. Vertical scanning does this. But in this case vertical scanning is done only to the width of the line. One output of this phase is shown in Figure 2.3.

Fig. 2.3 (a) Segmented word, (b) and (c) word after removing shirorekha

2.2.3 Segmenting Individual Characters
Once we get the words we segment it to individual characters. Before segmenting words to individual characters, we locate the header line or shirorekha. This is done by finding the rows having maximum number of black pixels in a word. Here we can use some sort of heuristic approach because some time all rows of shirorekha do not contain same number of black pixels. In this paper we assume that the maximum variation in the number of black pixels in each shirorekha is 10. This number may not be fixed for all script. This value may be chosen by hit and trial method. After locating shirorekha we remove it i.e. converts it in white pixels. After removing shirorekha our word is divided in three horizontal parts known as upper zone, middle zone and lower zone. Individual characters are separated from each zone by applying vertical scanning. Here vertical scanning is performed to the width of zones and length of the word. As noted earlier only modifiers are present in upper and lower zone. So before performing vertical scanning in these zones, it is checked whether any modifier exists or not. This is required because in several cases words may contain only middle zone or middle and upper zone or middle and lower zone only as shown in Figure 2.4.

Fig. 2.4 (a) Word having only middle zone and (b) word having all three zones, (c) Word having middle and upper zone (d) word having middle and lower zone.

3 CLASSIFICATION

In the present section we discuss the problem of recognition of the characters. We have investigated two classifiers. The first one is based on KL Divergence principle and second one uses Artificial Neural Network soft computing tool. Classification is performed based on the extracted features. We calculate feature vector for all Hindi character sets using three methods. We will first discuss the features and methods for their extraction and then describe the KL Divergence and ANN approach.

3.1 Feature Extraction
Feature extraction is one of the most important steps in developing a classification system. In selecting features for a classification system, it is necessary to study the characteristics of the script being classified. This would help in selecting features that better discriminate the character sample.

For initial classification of characters, we consider three features as follows:
1. Histogram of projection based on Mean Distance
2. Histogram of projection based on pixel value.
3. Vertical zero crossing

These features are used to design the classifier where decision is taken on the basis of presence/absence of feature value to classify a particular character.

3.1.1 Histogram of projection based on Mean Distance
This feature vector work as follows:
I. Find the x and y position of each black pixel and add them to find X and Y;
II. Divide the both X and Y by number of black pixels to find a central point;
III. Calculate the distances of each black pixel from this central point and complete a matrix and normalize it;
IV. Binaries these distances using histogram.

3.1.2 Histogram of projection based on pixel value
I. Find outer boundary of character;
II. Scan it horizontally as well as vertically to find the number of black pixels;
III. Take vertical and horizontal projection;
IV. Normalize it.

3.1.3 Vertical Zero Crossing
Image of a character is treated as an array of pixels. White pixel is expressed by 1 and a black pixel by 0. Tracing the whole array column by column, number of transitions from black to white pixel is recorded for each column.
Once we have found the features of all segmented characters using above three methods, we use them as input for the neural network. All characters have unique feature vector based on which they are recognized.

3.2 Tools Used for Classifier
As mentioned earlier, two techniques have been used to develop the actual classifier. These are based on KL divergence principle and Artificial Neural Networks (ANN). We will now discuss both methods.

3.2.1 KL Divergence measure
KL divergence is fundamental concept of statistics. It is a measure of the likelihood ratio. It is expressed as the expected log-likelihood ratio. It basically measures the distance between two distributions p and q. From an information theoretic perspective the measure S(p,q) measures the information in p, given a priori information q. We can express the above as: given two probability mass functions p(x) and q(x), the Kullback-Leibler divergence (or relative entropy) between p and q is defined as [12]

\[ D(p|q) = \sum p(x) \log p(x)/q(x) \]

In this paper we have calculated feature vector for all Hindi character sets using above mentioned three methods. The extracted feature values are used for the training and the results are stored. When a new character is to be recognized, the feature vectors of these new characters are calculated and they are matched to the feature vectors obtained from the trained characters. A nearest neighbor classifier is used in this approach.

3.2.1.1 Drawback of KL Divergence Approach
This KL divergence method is most suitable for handwritten characters but it fails in the case of printed document or documents having different fonts. The main reason for this may be the varying thickness of fonts. We can, of course, apply thinning algorithm [11] on segmented characters before feeding it to the classifier. However, this would involve some extra computation and thinning algorithms have their own limits. To overcome this problem we used an ANN approach for classification.

3.2.2 ANN Approach
Neural network approach has been used for classification and recognition. Training and recognition phase of the neural network has been performed using conventional back propagation algorithm with two hidden layers. The architecture of a neural network determines how a neural network transfers its input into output. This transfer can be viewed as a computation.

3.2.2.1 Scaling
Since ANN receives input in fixed size, normalization process is required to change each segmented character to be in same image size. There are two major issues that we should consider at this stage – the suitable fixed size of input matrix for ANN and a method to transform from the original character to the normalized one. From our study of the characteristics of printed Hindi Character of upto size 16, we have found that the matrix size 48x57 can fit any Hindi character. An example is given in Figure 3.1 where we scale an image of an arbitrary size to a fixed size. As can be seen, the scaling process does not introduce distortions or loss of information.

![Original 22x30 image](image_1.png) / ![Scaled 48x57 image](image_2.png)

**Fig. 3.1** Normalization from random size to fixed size.

3.2.2.2 Training
A neural network maps a set of input to a set of outputs. This nonlinear mapping can be thought of as a multidimensional mapping surface. The objective of learning is to mould the mapping surface according to a desired response. We have used four layered perceptron with two hidden layers, one input and one output layer. In input layer we used 169 neurons, equal to the number of elements in feature vector. The first hidden layer has 250 and the second hidden layer consists of 200 neurons. Finally in output layer we used 77 neurons, equal to the number of character in training set. Selection of number of neurons in each layer is purely based on practical observation. As mentioned earlier, we adopted back propagation for learning. This is the most popular learning algorithm in multilayer network [2]. It employs descent method in weight space. Thresholding is done by using sigmoid and tansig function. For the output of first hidden layer, we used the sigmoid function and at the second hidden layer, we used tansig function. We have trained our neural network for fives set of fonts each having 77 characters.

The list as given in Figure 3.3 shows the actual values obtained with training sample Ka after the completion of training. As can be seen, the first value is close to 1 while all the others are near to zero ( \(< 0.06\)). This agrees quite well with the expected values of the character.

\[ \text{value} = 0.9550, 0.0011, 0.0000, 0.0578, 0.0027, 0.0025, 0.0250, 0.0012, 0.0002, 0.0004, 0.0116, 0.0001, 0.0000, 0.0328, 0.0005, 0.0032, 0.0026, 0.0001, 0.0004, 0.0020, 0.0000, 0.0000, 0.0064, 0.0000, 0.0011, 0.0122, 0.0012, 0.0012, 0.0000, 0.0000, 0.0000, 0.0001, 0.0013, 0.0010, 0.0008, 0.0016, 0.0382, 0.0049, 0.0013, 0.0002, 0.0004, 0.0022, 0.0004, 0.1386, 0.0001, 0.0083, 0.0065, 0.0219, 0.0143, 0.0000, 0.0135, 0.0002, 0.0007, 0.0022, 0.0005, 0.0022, 0.0036, 0.0183, 0.0022, 0.0003, 0.0016, 0.0037, 0.0001, 0.0129, 0.0001, 0.0089, 0.0065, 0.0008, 0.0000, 0.0028, 0.0004, 0.0018, 3.0003, 0.0074, 0.0045 \]

**Fig. 3.3** Values of all 77 characters

3.2.2.3 Efficiency
In neural network time to converge is highly dependent on the learning parameter [9]. If it is too high, then the training does not converge. On the other hand, if learning
parameter is assumed very low, then it will take excessive
time to converge. Adjusting the learning parameter
dynamically during the process of training has given
better result. Starting with a higher value, we decrease the
parameter as it approaches convergence. In our case first
we trained using 0.9 as the learning parameter and then
we reduced it to 0.1. As mentioned in previous paragraph,
we trained the neural network for five fonts having 77
characters each. So there are 385 total characters in the
training set. Each character is represented by a feature
vector of length 169. So, the amount of training data is
very huge data. After training for several hours and lacks
of epochs it was able to train 383 characters. i.e. only two
are not trained. So we can say that efficiency of training
of our neural network is almost 100%.

4 RESULTS

We tested our OCR on various printed documents and
gathered various results. The performance, at the level of
individual characters, is excellent with a correct
recognition rate of over 99%. However, at the level of
words in a text the performance drops and an accuracy of
90% is obtained. One of the samples which we used for
testing the performance of our OCR is given in Figure
4.1(a). The results of the text obtained after recognition
is shown in Figure 4.1(b). As noted earlier, the recognition
rate is superior for isolated characters than for continuous
characters. The discrepancy in performance in case of
continuous characters may have happened due to the
errors in the segmentation procedure. We now discuss the
results in detail.

4.1 Robustness of Classifier

We first demonstrate some results of the classifier at
class level. To observe the robustness of the classifier
we gradually increase the distortion in the image and
observe the recognized character. In Table 4.1, we have
given the original character, the distorted character given
as an input to the classifier, as well as the character
recognized by the classifier. From an examination of this
table, it is clear that our classifier is very robust and is
able to correctly classify highly distorted characters.

4.2 Feature Extraction Techniques

From the results presented above, we conclude that the
error due to character classification is extremely small.
This is, indeed, due to a good choice of the features. To
appreciate this point better we next present

| Table 4.1 |
| Original Character | Distorted Character | Character recognized |

---

some detailed results of the feature values obtained, for
several characters. We have used a set of robust and
efficient feature extraction techniques. These feature
extraction techniques are independent of fonts and its
size. The classifying effect we have seen in earlier
paragraph. Now we will see the feature diagrams for the
Devanagari character (V) in Figure 4.2 to Figure 4.6.

Similarly we can draw feature diagram for every
class. We get figures, which shows that each
class has unique feature values. It indicates the
discriminating power of these features and is the primary
reason for the robustness of the recognition.
5 CONCLUSION

This experiments have illustrated that the artificial neural network concept can be applied successfully to solve the Hindi Optical Character Recognition problem. There are many variations of factors that affect the performance of the developed Hindi OCR software. We conclude at this moment that the input matrix in size 48x57 gives better result than others choices. The recognition rate of the OCR with the real Hindi document is quite high as shown in the result part. However, other kinds of preprocessing and neural network models may be tested for a better recognition rate in the future research in our current Hindi OCR. Our character segmentation method could be improved to handle larger variety of touching characters that occur often in images obtained from inferior-quality documents. The test set used in this experiment is of 77 characters of five different types of fonts. This can be increased. The toughest phase in the experiment is getting a good set of characters for classification. Also, the characters used for experiment were enclosed in the bounding region of a fixed size. Although this may not be the always case, attention is drawn towards the fact that mere shape based recognition will not always perform well. Especially when the system is dealing with ligatures. We can attach speech synthesizer with our OCR with an aim to make a system for reading aid to the blind.

The other future enhancement that can be done on this paper is to use a dictionary of words to correct the output [13]. Certainly this will improve the performance. We have implemented this for script having only Devnagari characters. We can extend it to classify the document having more than one script characters.

REFERENCE