Available online at www.isas.org.in/jisas
JOURNAL OF THE INDIAN SOCIETY OF
AGRICULTURAL STATISTICS 72(1) 2018 77–82

A New Approach using Template Matching for Recognition of Handwritten Odia Text

Sachikanta Dash[1] and Sukanta Dash[2]
[1]*Maharaja Institute of Technology, Bhubaneswar*
[2]*ICAR-Indian Agricultural Statistics Research Institute, New Delhi*

Received 09 November 2017; Accepted 21 March 2018

SUMMARY

The objective of Handwritten Optical Character Recognition (HOCR) is automatic reading of optically intellect document text materials to translate human-readable characters to machine- readable codes. In Optical Character Recognition, the text lines in a document must be segmented properly before recognition. English Character Recognition (CR) has been extensively studied in the last half century and progressed to a level, sufficient to produce technology driven applications. But same is not the case for Indian languages which are complicated in terms of structure and computations. This is the motivation behind choosing OCR for Odia language. An odia handwritten paragraph is chosen for processing and recognition. The HOCR system is devised to first segment the whole document into text lines, then to words and then to individual characters. These characters are then used to extract the necessary features and recognize those characters and classify them.

Keywords: Optical Character Recognition (OCR), Handwritten Odia Character Recognition (HOCR), Template matching.

1. INTRODUCTION

Optical character recognition has many different practical applications. The important area where OCR has vast applications includes text entry in office automation system, Banking sector data entry facilities and many more. The present state of the art in OCR has moved from primitive schemes for limited character sets, to the application of more sophisticated techniques for Omni font and handprint recognition. Great numbers of algorithm have been developed for character recognition. The problem is not yet solved satisfactorily, especially not in the cases when there are no strict limitations on the handwriting or quality of print.

Handwritten character recognition (HCR) is a technique by which a computer system could recognize characters and other symbols written in natural handwriting. Early HCR techniques were based on template matching, simple line and geometric features, stroke detection and their derivatives extraction (Sharma and Jha, 2015). Handwritten documents are difficult to be processed because they lack a specific structure. As exhaustive research and development on HCR went by and with several conferences and workshops, modern techniques advanced rapidly. The subject of HCR gained considerable momentum and grew swiftly. As of now, many new algorithms and techniques in preprocessing, segmentation and classification have been developed.

Given the testing sample which is a scanned handwritten document where each character needs to be recognized. In the initial stages, choosing proper and suitable preprocessing and segmentation algorithms the characters are extracted. Efficient feature extraction and classification methods should be used to maintain good performance and accuracy of results.

2. PROPOSED METHODOLOGY

The proposed work aims at providing a customized template matching approach with the improved

Corresponding author: Sukanta Dash
E-mail address: sukanta.iasri@gmail.com

performance for recognition of the handwritten odia character set. To improve the performance of the template matching algorithm a threshold measure called SIMILARITY MEASURE is imposed along with the correlation analysis between test and trained templates. Further, to simplify the process of classification a database is incorporated. The architecture of the proposed system for handwritten odia character recognition is represented in Fig. 1.

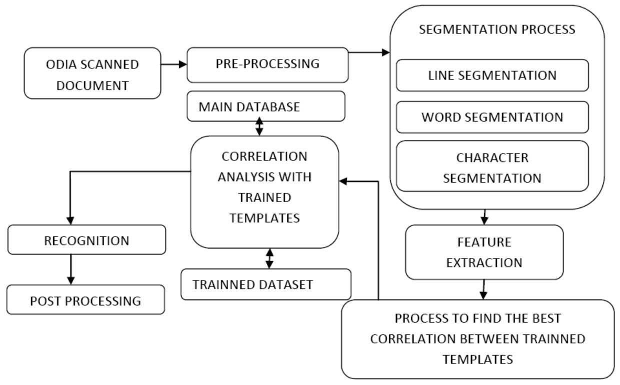


Fig. 1. Architecture of Handwritten Odia character Recognition

3. PREPROCESSING

This step is done to enhance the document by removing noise and other distortions in the written material. This step includes skew detection and noise removal (Fig. 2). When a document is fed to the optical sensor either (scanner) mechanically or by a human operator to get the digital image, a few degrees of skew (tilt) is unavoidable. Skew angle is the angle that the text lines in the digital image makes with the horizontal direction. Skew estimation and correction are important preprocessing steps of line and word segmentation approaches. Skew correction can be achieved by (i) Estimating the skew angle, and (ii) Rotating the image by the skew angle in the opposite direction. There are three main reasons for removing the skew. The first is appearance, where anything more than about 0.25 degrees (0.004 radian) is quite noticeable. The second is that it is important to remove skew if any analysis is to be done on the page. The presence of skew, and particularly more than 0.01radian, complicates the analysis of page elements such as text columns. The third is that the performance of symbol based compression on multipage documents is badly degraded by random skew of 0.01radian or more, because the same characters are placed in different equivalence classes due to skew.

The methods for determining skew are:

- Use all the pixels
- Use projection profiles
- Use the variance of the projection profiles
- Use the variance of the projection profile derivative
- Use linear and binary search

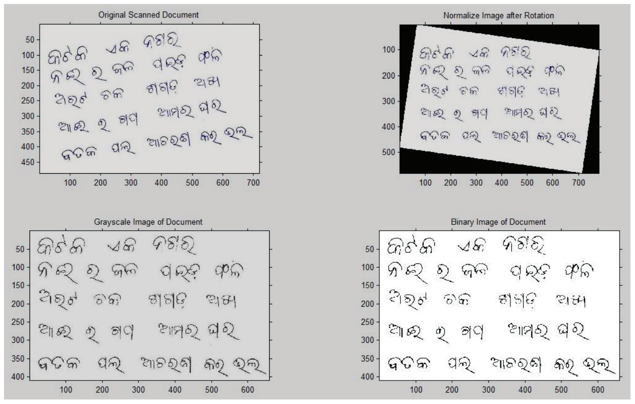

Original Scanned Document
Normalize Image after Rotation
Grayscale Image of Document
Binary Image of Document

Fig. 2. Preprocessing

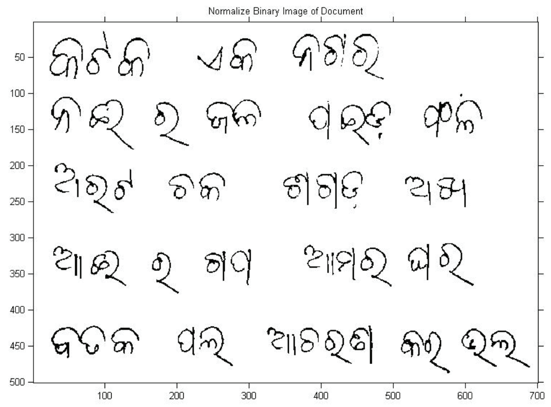


Fig. 3. Normalized Binary Document

A projection profile based method is being implemented. A straightforward solution to determining the skew angle of a document image uses a horizontal projection profile. This is a one-dimensional array with a number of locations equal to the number of rows in an image. Each location in the projection profile stores a count of the number of black pixels in the corresponding row of the image. This histogram has the maximum amplitude and frequency when the text in the image is skewed at zero degrees since the number of co-linear black pixels is maximized in this condition. The peaks in the profile calculated from the deskewed image (Fig.2) shown in are taller and more closely spaced than those computed from the skewed image shown in (Fig. 2). This characteristic has been used in several algorithms. One way of doing this is to rotate the input image through a range of angles and calculate the projection profile for each one. Finally a normalized binary (Su and Tan, 2013) scanned document is generated for further processing (Fig. 3 & Fig. 4).

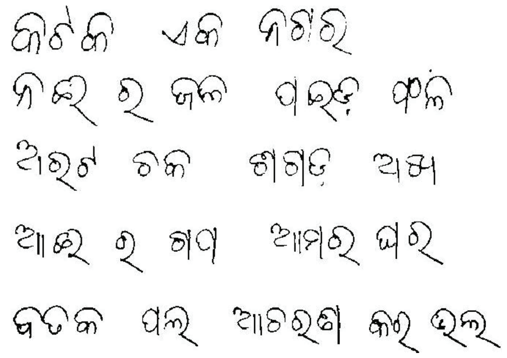


Fig. 4. Normalized Binary document

4. SEGMENTATION

The scanned document is at first passes through the preprocessing (like de-noising, skew correction etc) block prior to segmentation. Segmentation decomposes the document image into subcomponents like lines, words and characters. To achieve greater accuracy, segmentation and recognition could not be treated independently. Most of the existing line segmentation methods have limitations when applied to unconstrained handwritten documents. In case of segmentation, at first lines are separated then from lines words are separated and finally the characters are separated from words.

4.1 Line Segmentation Methods

A cleaned image obtained after preprocessing is the input for segmentation, which decomposes the image into subcomponents like lines, words and characters based on various segmentation approaches (Kumar *et al.*, 2012) Cursive handwritten texts are segmented using more advanced methods like Hidden Markov Model, Artificial Neural Networks, linear programming, genetic programming and contextual methods. The existing methods of text line segmentation could be grouped into two classes: top-down approach and bottom-up approach. Top-down methods start from the whole image and iteratively subdivide into smaller blocks to isolate the components. Bottom-up methods group small units of image (pixels, connected components, etc.) into text lines and then text regions. Most of the existing line segmentation methods have limitations when applied to unconstrained handwritten documents (Pal *et al.*, 2009 & Zanaty, 2012) because they more or less assume horizontal, straight, parallel and untouched text lines. Methods based on connected components are faster, but suffer from touching or close proximity of components (Mohiuddin and Mao, 2014 & Singh *et al.*, 2014). A few methods like projection profile, Hough transform and smearing failed to segment the text lines properly in case of very closely spaced lines. Some additional techniques were required as post processing steps to isolate touching text lines. The global horizontal projection method is used to compute sum of all white pixels on every row and construct corresponding histogram. The steps for line segmentation are as follow:

- Construct the Horizontal Histogram for the image.
- Count the white pixel in each row.
- Using the Histogram, find the rows containing no white pixel.
- Replace all such rows by 1.
- Invert the image to make empty rows as 0 and text lines will have original pixels.
- Mark the Bounding Box for text lines using standard Matlab functions.
- Copy the pixels in Bounding Box and save in separate file. (lines shown in Fig. 5).

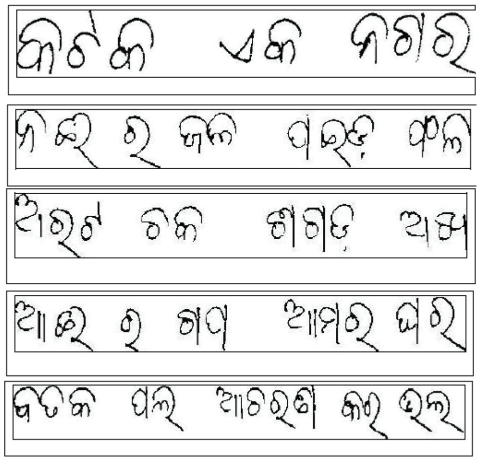


Fig. 5. Line Segmentation of document

4.2 Word Segmentation

After a text line is segmented, it is scanned vertically, column by column. If a column contains two or fewer black pixels, the scan is denoted by 0, else it is denoted by the number of black pixels in that column. In this way, a vertical projection profile is constructed. Now, if in the profile there exists a run of at least k consecutive 0s, the midpoint of that run is considered the boundary between two words. The value of k is taken as two-thirds of the text line height (text line height is the normal distance between the mean line and the base line).

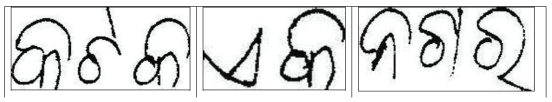


Fig.6. Word segmentation of document

In word segmentation method, a text line is taken as an input. After a text line is segmented, it is scanned vertically.

- Column wise dissection: (vertical projection)
- Calculate the sum of pixels column-wise.
- Obtain the column-wise projection of the inverted sub-images
- Find the minimum points from these projections
- A pair of consecutive minimum points defines one segmentation boundary.

Since the words do not touch each other word segmentation will not pose issues.

In the first step the distance between adjacent characters in the text line image are computed. In the second step the computed distances are classified as either inter-word distances or inter-character distances using threshold value. The distances between words are always larger than distances between characters. Words can be segmented by comparing the distances with a suitable threshold. The threshold is defined as

$$Threshold = \frac{\text{Sum of the distances of adjacent characters}}{\text{Number of distances}}$$

When the distance value is greater than the threshold it is treated as a word gap otherwise it is a character gap. Words are segmented using the word gap (Separated words are shown in Fig. 6).

4.3 Character Segmentation

A slight modification in previous algorithm (section 4.2) is used here. The steps for line segmentation are as follow:

- Get the thinned image using Matlabbwmorph function. (This is done to normalize image against thickness of the character).
- Count the white pixel in each column.
- Find the position containing single white pixel.
- Replace all such columns by 1.
- Invert the image to make such columns as 0 and text characters will have original pixels.
- Mark the Bounding Box for characters using standard Matlab functions, see Fig 7.
- Copy the pixels in the Bounding Box and save in separate file. (Separated characters are shown in fig. 7).

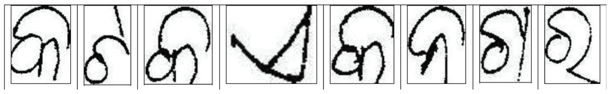


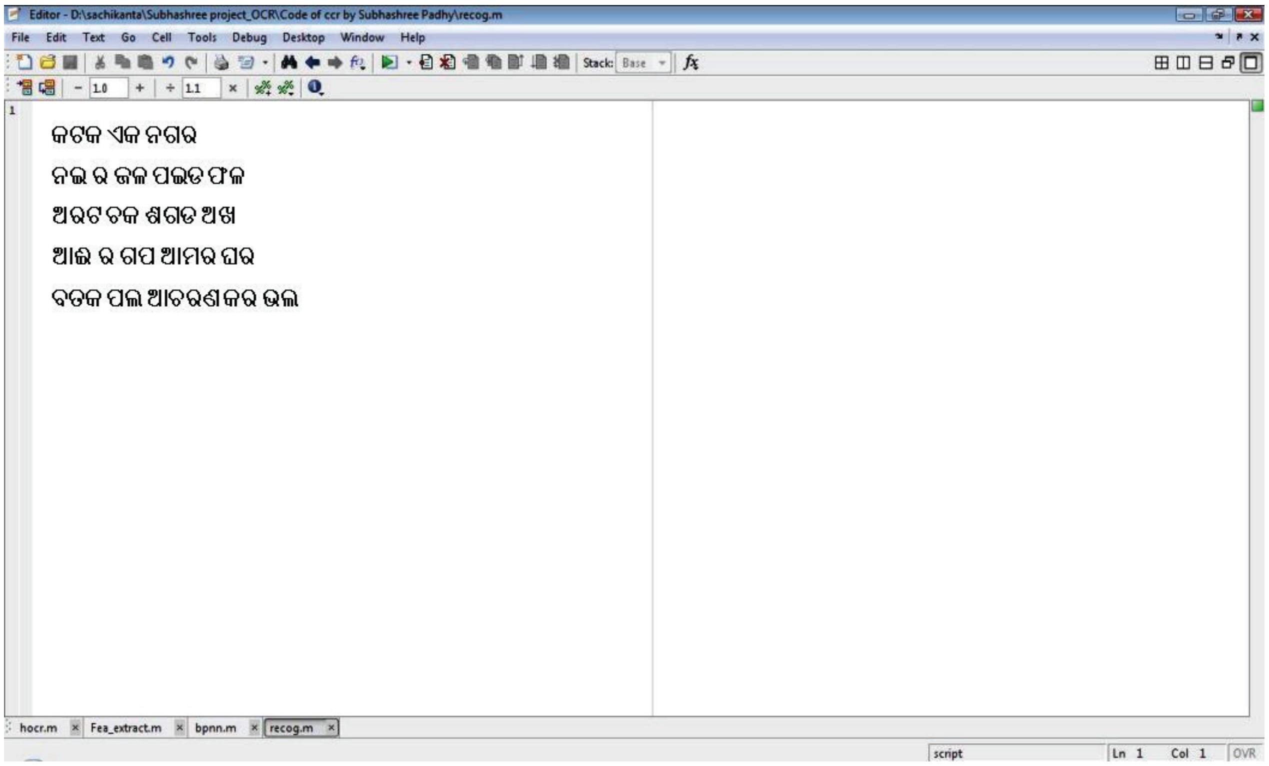
Fig. 7. Character segmentation of document

5. OPTICAL CHARACTER RECOGNITION

To recognize the characters from the input image is carried out using the Template Matching method. After the character wise segmented images are received, matching (Bengio *et al.*, 2013) those with the templates of the dataset identifies the characters. This is done with the help of 'corr2' function of MATLAB. The concept of this function is to detect similarities in 2D patterns with cross correlation method. Here, the input image is stored as matrix in A_{mn} while compared against a template, which is in B_{mn}. The return value 'r' indicates the match ability between the input image and template. After all comparisons, the highest correlation coefficient value of 'r' is identified as the lower case letter or selected special character.

'corr2' function in MATLAB computes the correlation coefficient using the equation as follows:

corr2 computes the correlation coefficient using


Editor - D:\sachikanta\Subhashree project_OCR\Code of ccr by Subhashree Padhy\recog.m
File Edit Text Go Cell Tools Debug Desktop Window Help
hocr.m
Fea_extract.m
bpnn.m
recog.m
script
Ln 1
Col 1
OVR

```
କଟକ ଏକ ନଗର
ନଈ ର ଜଳ ପଈଡ ଫଳ
ଅରଟ ଚକ ଶଗଡ ଅଖ
ଆଈ ର ଗପ ଆମର ଘର
ବତକ ପଲ ଆଚରଣ କର ଭଲ
```

Fig. 8. Output of Recognized Document

$$r = \frac{\sum_m \sum_n (A_{mn} - \bar{A})(B_{mn} - \bar{B})}{\sqrt{\left(\sum_m \sum_n (A_{mn} - \bar{A})^2\right)\left(\sum_m \sum_n (B_{mn} - \bar{B})^2\right)}}$$

where $\bar{A}$=mean(A), and $\bar{B}$=mean(B).

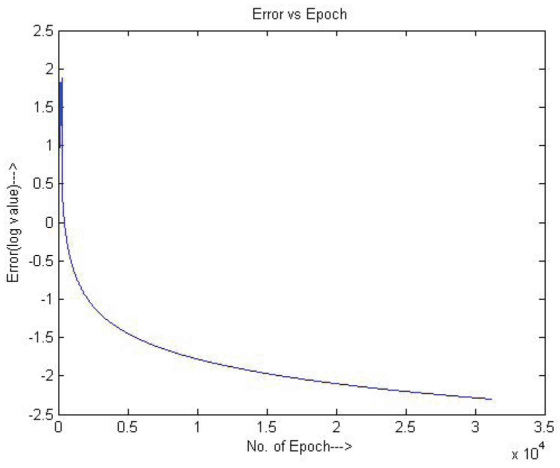


Fig. 9. Error vs Epoch report

The result obtained in generated text file (Fig. 8) is correct owing to the line and character wise segmentation method, which starts to read the input from the first line of the texts to the bottom line and then from left side to the right side. The error versus epoch report is shown in Fig. 9. The advantages of using this method is owing to its simplicity but can be used for recognizing (Bhattacharya and Chaudhuri, 2005) various sets of handwriting provided through templates.

The major factor that affected the accuracy of handwritten (Pal *et al.*, 2007 & Bhattacharya and Chaudhuri, 2009) code was the problem in mismatch in templates and the character of input image. The dataset of the templates was comparably small and was not able to handle variations in similar looking handwritten letters such as 'ଣ', 'ଶ', 'ଗ' and 'ଖ'. These caused problem in template matching as it was often mismatched with 'କ' and 'ଳ' of the dataset. But, considering the all factors involved it is an easy and efficient method to implement and learn the concept of machine learning.

6. CONCLUSION AND FUTURE SCOPE

The handwritten Odia Character recognition process using template matching has been implemented successfully. The importance of pre-processing phase in this project is immense and includes steps like Skew detection and correction, binarization, removal of smaller components like noise removal. The process of template matching thus can be summarized as comparing the input image with a stored template and identify the character from the input image according to its highest match using coefficient of correlation. Through this method, handwritten text is recognized with good accuracy. This experiment with 100% accuracy. The scope for future work is enormous. Inclusion of advanced image processing techniques like skew correction, adjustments for lighting in image could be done. The accuracy of the project can be improved when it can be expanded to include a wide variety of handwritings and bigger set of handwritings which enables us to encompass large set of variations. With inclusion of larger data sets and handwriting variation, advanced techniques like neural networks could also be implemented. As the proposed system do not uses compound odia characters, future scope is there to implement this process in compound handwritten odia characters. The experimental results of the proposed system can be still improved by replacing template matching with an efficient feature extraction technique so as to reach high recognition accuracies.

REFERENCES

Bhattacharya, U. and Chaudhuri, B.B. (2005). Databases for research on recognition of handwritten characters of Indian scripts, *8th International Conference on Document Analysis and Recognition*, 789-793.

Bhattacharya U. and Chaudhuri B.B. (2009). Handwritten numeral databases of indian scripts and multistage recognition of mixed numerals, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **31(3),** 444-457.

Bengio,Y., Courville, A., and Vincent, P. (2013). Representation learning: A review and new perspectives, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **35(8),**1798-1828.

Mohiuddin, K. and Mao, J. (2014). A comparative study of different classifiers for handprinted character recognition, *Pattern Recognition in Practice IV*, 437-448.

Pal, U., Wakabayashi T. and Kimura F. (2009). Comparative Study of Devnagari Handwritten Character Recognition using Different Feature and Classifiers, *Proc. 10th International Conference on Document Analysis and Recognition*, 1111-1115.

Pal, U., Sharma, N., Wakabayashi, T. and Kimura, F. (2007). Handwritten numeral recognition of six popular indian scripts, 9th *International Conference on Document Analysis and Recognition*, **2**, 749-753.

Sharma, A. and Jha, S.K. (2015) "Identification of alphanumeric patterns using android," *International Journal on Recent and Innovation Trends in Computing and Communication*, **3(4),** 2455-2470

Singh, P.K., Sarkar, R., Das, N., Basu S. and Nasipuri M. (2014). Statistical comparison of classifiers for script identification from multi-script handwritten documents. *International Journal of Applied Pattern Recognition*, **1(2),** 152-172.

Su, B., Lu S., and Tan, C.L. (2013). Robust document image binarization technique for degraded document images, IEEE Transactions on Image Processing, **22(4),** 1408-1417.

Zanaty, E.A. (2012). Support Vector Machines (SVMs) versus Multilayer Perception (MLP) in data classification, *Egyptian Informatics Journal*, **13**, 177-183.