IDENTIFICATION AND SEGMENTATION OF TOUCHING BRAHMI CHARACTERS FROM DEGRADED DIGITAL ESTAMPAGE IMAGES USING ENSEMBLE CLASSIFIER

Aniket S. Nagane

Assistant Professor, MAEER's MIT-Arts, Commerce and Science College, Alandi, Pune 412105, India
Department of Computer Science, Savitribai Phule Pune University, Pune 411007, India;
asnagane@mitacsc.ac.in; aniket.nagane@gmail.com

Shankar M. Mali

Professor, Dr. Vishwanath Karad, MIT World Peace University, Pune India;
Department of Computer Science, Savitribai Phule Pune University, Pune 411007, India
shankar.mali@mitwpu.edu.in; shankarmali007@gmail.com

Abstract

Brahmi script is an ancient Indian script that requires expertise to read the written documents. OCR system can help reading and understanding the documents of Brahmi script. In the OCR systems, to recognize the symbols, accurate segmentation in earlier phase is required. If the segmentation goes wrong, it indicates the failure of the recognition process. Identification and segmentation of touching characters is a challenging task in OCR systems. In this paper, the authors have proposed an algorithm which successfully identifies and segments the touching characters from the Brahmi script documents. The algorithm uses ensemble classification technique to identify the touching characters using 9 unique features. The proposed algorithm has achieved 100% accuracy in identification and 99.16% accuracy in segmentation of touching Brahmi script characters. The results of proposed algorithm are significant to identify and correctly segment the touching Brahmi script characters from degraded digital estampage images.

***Keywords*: Brahmi script; estampage; OCR; segmentation; touching characters.**

1 Introduction

In traditional India, people used to believe more in verbal transfer of knowledge rather than the written communication. History of written communication in Indian culture is newer as compared to the classical ancient cultures of China and Japan (Ojha 1971). The Brahmi script is considered to be oldest Indian ancient script in Central Asia and Indian Subcontinent. Brahmi script originated in BC 400 and was practiced till the period of AD 500 (Salomon 1998).

Brahmi script was widely used during the period of emperor Ashoka from BC 268 to BC 232 when it spread across the Indian subcontinent and became popular. Brahmi script inscriptions are mostly found engraved on rocks, copper-plates (tampra-patra) and bhurja-patra (Fig. 1).

Fig. 1 Brahmi script inscriptions (a) rock inscriptions; (b) copper-plate (tamra-patra) inscription

Eventually, rock inscriptions of the Brahmi script are getting degraded day by day due to rock erosion. Archeological Survey of India (ASI), Mysore, India is the central government institute that preserves the inscriptions in the digital format by maintaining the glass negatives of estampages. Estampage is an impression of inscriptions on special paper (Fig. 2). The paper is inked from one side and is pasted on the inscription. Further, the impression is taken on the paper using slow wooden hammering (https://en.wikipedia.org/wiki/Estampage#:~:text=From%20Wikipedia%2C%20the%20free%20encyclopedia,paper%20for%20a%20clearer%20read [accessed on October 19, 2021]).

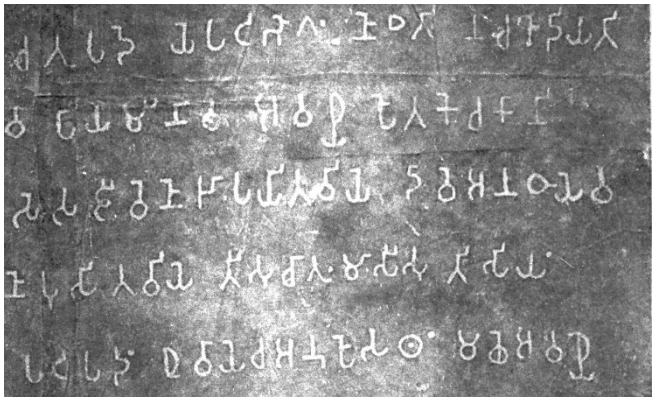

Fig. 2 Estampage of Brahmi script inscription (source: ASI, Mysore, India)

Considering the longer time span, diverse geographical areas, and usage of different mediums to write the Brahmi script, variations are observed in its appearance. The variations are found in the form of inconsistent character set, overlapping of text causing mixing of characters etc. Such Brahmi script documents are discovered even today. Only the experts of Brahmi script and contemporaneous languages can read such documents. On the other hand, optical character recognition (OCR), can definitely help transliterate the Brahmi script document into a document containing a modern-day script (Garain *et al.* (2012); Jha and Parvathy (2019)). This will help anyone (those not having any expertise or knowledge about the Brahmi script) to read and understand the contents of the document

OCR systems are used to digitalize the documents available in hard copy (Hamad and Kaya (2016); Ullah *et al.* (2019); Zho *et al.* (2016)). Hard copy documents can be categorized as (i) Historical Documents; (ii) Handwritten Documents; and (iii) Printed Documents. These documents may contain one or several scripts (multilingual document), special symbols, images, or icons. OCR system converts these documents into a digitalized document with same contents (Karthick *et al.* (2019)). OCR system recognizes the engraved, handwritten, or printed characters from scripts' character set and generates the equivalent characters in digitalized form. In OCR systems, some operations are performed in a sequence to get the results. Some important operations are (i) Image Acquisition; (ii) Pre-processing; (iii) Segmentation; (iv) Feature Extraction; and (v) Classification and Recognition (Farulla *et al.* (2017); Mittal and Garg (2020)). Image acquisition is the very first and basic step of OCR system, in which, using some source we can capture an image of the object, wherein OCR objects are generally the hard-copy documents. Once the image has been captured, the next process is noise removal, image cropping, enhancement of the image as per the requirement of the further steps, etc. After this, the crucial step is segmentation of characters or symbols from the document. This segmentation can be done in the following steps: (i) Line Segmentation; (ii) Word Segmentation; and (iii) Character segmentation (Ali and Suresha (2019); Garain and Chaudhuri (2002); Peng *et al.* (2017); Rakshit *et al.* (2018)). The process of segmentation is a major operation in the OCR systems, where the results of next steps such as feature extraction and recognition are totally based on the accuracy of segmentation. Detail analysis of

segmentation, lists the problems that cause in-accurate segmentation (Burra *et al.* (2018); Deshmukh and Kolhe (2019); Jain (2017); Ramalingam and Bhojan (2018)). After the accurate segmentation, feature vector is generated for every individual character. Using this feature vector, correct labelling of the characters is performed and recognition phase is completed.

2 Analysis of Segmentation

Segmentation techniques are mainly based on connectivity analysis of the objects in the image. A unique connected object is considered as a single character. When the two pixels in a digital image are not connected with each other, they are considered to be a part of two different objects. The historical documents contain problems like bleed through effect, broken characters, overlapping characters, touching characters, etc. In case of two-sided documents, over the years, the ink from one side of the page is visible on the other side which gives the bleed through effect. Over a long period, the ink on the papers fade out or the degradation of the papers may lead to broken characters. Overlapping characters are recognized when the strokes of characters from two adjacent rows are mixed into each other. Touching characters are similar to the overlapping ones where the strokes of adjacent characters in the same row get mixed. Here, mixing of strokes means characters have connected pixels (Patil and Mali (2015); Rehman and Saba (2011); Roy *et al.* (2009); Saba *et al.* (2011); Saba *et al.* (2014); Sen *et al.* (2018)).
In all the above cases, connectivity analysis-based segmentation does not work appropriately. In case of the bleed through effect, unnecessary shapes connect to the objects. In case of broken characters, single object gets segmented as two or more number of objects, and in case of overlapping and touching characters, two or more characters get segmented as one single character. In this paper, we have proposed the method to segment the touching characters of Brahmi script.

3 Data Collection

To implement the proposed method and test the results, we have created our image set consisting 8014 characters as no standard image set is available for Brahmi script characters. We have acquired all the images from Archeological Survey of India (ASI), Mysore, India and Bhandarkar Oriental Research Institute, Pune. By using these images, we have performed pre-processing and segmentation and have generated an image set of 8014 characters, including the touching characters. The same image set is used for identifying and segmenting the touching characters.

4 Proposed Methodology

The proposed method is implemented in the following parts: identification of touching characters and identification of segmentation points. In the first part of the algorithm, we identify touching and non-touching characters using ensembled classifier. Second part of the algorithm covers segmentation of touching characters using size of the characters and their aspect ratio with each other.

4.1 Identification of touching characters using ensemble classifier

Four different classification techniques viz. Ensemble classification using boosting algorithm, Support Vector Machine with Error correcting output codes, K-Nearest Neighbour and K-means clustering are implemented on the dataset. Ensemble classifier has produced better results comparatively, i.e., identification of touching characters is done using ensembled classification algorithm. Ensemble classification is a process where outcomes of all the present models are analysed and combined together to form a new model so as to acquire better results as compared to the individual models. While using individual classification models, some of the features from the feature set may not contribute to the final result, and hence, implementing the ensemble classification technique makes the use of all the features to improve the accuracy (Liu and Zhang (2019)).
Ensemble Classifier uses boosting algorithms for classification. AdaBoostM1 is a boosting algorithm for binary classification and AdaBoostM2 is an extension of AdaBoostM1 for multiple classes. The boosting algorithm trains learners sequentially. For every learner with index *t*, AdaBoostM1 computes the weighted classification error where as AdaBoostM2 uses weighted pseudo-loss for *N* observations and *K* classes

$$\varepsilon_t = \frac{1}{2}\sum_{n=1}^{N}\sum_{k\neq y_n} d_{n,k}^{(t)}(1-h_t(x_n,y_n)+h_t(x_n,k)), \quad (1)$$

where

- $h_t(x_n,y_n)$ is the confidence of prediction by learner at step *t* into class *k* ranging from 0 (not at all confident) to 1 (highly confident).
- $d_{n,k}^{(t)}$ are observation weights at step *t* for class *k*.
- y_n is the true class label taking one of the *K* values.

- The second sum is over all classes other than the true class y_n.

The default learning rate for boosting algorithms is 1. When the learning rate of an algorithm is a lower number, its learning rate is also slow, however, when the learning rate is slow, it converges a better solution. Learning at a rate less than 1 is often called "shrinkage".
In AdaBoostM2, classification accuracy is measured as Pseudo-loss. Pseudo-loss behaviour can be explained as the first few learners in a boosted ensemble give low pseudo-loss values. After the first few training steps, the ensemble begins to learn at a slower pace, and the pseudo-loss value approaches 0.5 from below.

We also have implemented individual classification models which are KNN, SVM, and K-means clustering.

- K-nearest neighbour is supervised machine-learning algorithm used for classification. The behaviour of this classification model when the dataset or weights contain missing observations can be described as below:

  If any value of class labels(Y) or any weight is missing, then it removes those values from class labels, the weights, and the corresponding rows of predictor data(x) from the data. The software renormalizes the weights to sum to 1.
  When standardized predictors are specified without scale, classification model removes missing observations from individual predictors before computing the mean and standard deviation.
  When standardized predictor is specified with Prior or Weights, then the observations weights are also considered. Specifically, the weighted mean of predictor j is

$$\overline{x_j} = \sum_{B_i} w_k x_{jk} \qquad (2)$$

  and the weighted standard deviation is

$$s_j = \sum_{B_i} w_k (x_{jk} - \overline{x_j}) \qquad (3)$$

  where B_j is the set of indices k for which x_{jk} and w_k are not missing.
  When the Mahalanobis distance is specified without scale or covariance matrix, classification model removes rows of X that contain at least one missing value. In this case, the model does following things:
    (1) Computes the means and standard deviations of each predictor
    (2) Standardizes the data using the results of step 1
    (3) Computes the distance parameter values using their respective default.

  If you specify Scale and either of Prior or Weights, then the software scales observed distances by the weighted standard deviations.
  If you specify Cov and either of Prior or Weights, then the software applies the weighted covariance matrix to the distances. In other words,

$$Cov = \frac{\sum_B w_j}{\left(\sum_B w_j\right)^2 - \sum_B w_j^2} \sum_B w_j \left(x_j - \overline{x}\right)' \left(x_j - \overline{x}\right) \qquad (4)$$

  where B is the set of indices j for which the observation x_j does not have any missing values and w_j is not missing.

- Support Vector Machines (SVMs) are a set of supervised learning methods. In general, SVMs are used for binary classification problems. By using Error-Correcting Output Codes Technique, we can redesign multiclass classification problem as multiple binary classification problems. We have used SVM with ECOC and its model can be designed as below:

  Let M be the coding design matrix with elements m_{kl}, and s_l be the predicted classification score for the positive class of learner l. The algorithm assigns a new observation to the class ($\hat{k}$) that minimizes the aggregation of the losses for the L binary learners.

$$\hat{k} = \arg\min_{k} \frac{\sum_{l=1}^{L} | m_{kl} | g(m_{kl}, s_l)}{\sum_{l=1}^{L} | m_{kl} |} \quad (5)$$

ECOC models can improve classification accuracy, compared to other multiclass models.

- *k*-Means Clustering is unsupervised machine learning algorithm used to group datapoints with similar properties. *k-means clustering* is an iterative, data-partitioning algorithm that assigns *n* observations to exactly one of *k* clusters defined by centroids, where *k* is chosen before the algorithm starts.

  The algorithm proceeds as follows:
  (1) Choose *k* initial cluster centers (*centroid*). For example, choose *k* observations at random.
  (2) Compute point-to-cluster-centroid distances of all observations to each centroid.
  (3) Compute the average of the observations in each cluster to obtain *k* new centroid locations.
  (4) Repeat steps until cluster assignments do not change, or the maximum number of iterations is reached.

For classification, we have to identify unique features from the image. We have identified 9 such features to distinguish the image as touching or non-touching character (Fig. 3). All features are described below

(1) AF: It is a feature which describes the area of an object. Let I is an image defined as f(i,j) and J is an image which is a subset of I determined as 8 connected component from the image then number of pixels in J is indicates the area of J
AF = |A| where A is collection of pixels belonging to J
(2) CAF: Convex area feature is a numeric value indicating the number of pixels in convex hull C_h of the object.
CAF = |C| where C is collection of pixels belonging to C_h
(3) PF: Perimeter feature is the numeric value computed as number of pixels lying around the border B of object region.
PF = |P| where P collection of pixels belonging to B
(4) MAL: Major Axis length feature is a scalar value which describes number of pixels forming the major axis 'M' of the image ellipse.
MAL = |M| where M is major axis of the image ellipse
(5) MnAL: Major Axis length feature is a scalar value which describes the number of pixels forming minor axis 'Mn' of the image ellipse.
MnAL = |Mn| Where Mn is minor axis of the image ellipse.
(6) Cnt1: It is the numeric value indicating number of spikes (high) in a vertical projection profile of first horizontal part of an image.
Cnt1 = |T|
T={x / x ϵ vp & x<=1}
Vp = {y / y = sm}
Sm(i) = $\sum_{r=1,c=i}^{n/2} f(r,c)$
where n is number of rows, i=1to m and m is number of columns
(7) Cnt2: It is the numeric value indicating number of valleys (low) in a vertical projection profile of first horizontal part of an image
Cnt2 = |T|
T={x / x ϵ vp & x>1}
Vp = {y / y = sm}
Sm(i) = $\sum_{r=1,c=i}^{n/2} f(r,c)$
where n is number of rows, i=1to m and m is number of columns
(8) Cnt3: It is the numeric value indicating number of spikes (high) in a vertical projection profile of second horizontal part of an image.
Cnt3 = |T|
T={x / x ϵ vp & x<=1}
Vp = {y / y = sm}

$$Sm(i) = \sum_{r=n/2+1,c=i}^{n} f(r,c)$$

where n id number of rows, i=1 to m and m is number of columns

(9) Cnt4: It is the numeric value indicating number of valleys (low) in a vertical projection profile of second horizontal part of an image

Cnt3 = |T|

T={x / x ϵ vp & x>1}

Vp = {y / y = sm}

$$Sm(i) = \sum_{r=n/2+1,c=i}^{n} f(r,c)$$

where n id number of rows, i=1 to m and m is number of columns

Using these features, we implemented ensembled classification technique to identify touching characters. Table 1 describes sample of features for touching and non-touching characters.

Type of character	AF	CAF	MAL	MnAL	Peri	cnt1	cnt2	cnt3	cnt4
Touching	381	1299	63.62592	44.46662	289.73	34	20	4	51
	417	1167	48.70971	43.76582	214.611	20	20	2	35
	467	1054	85.21813	37.86985	229.929	6	36	13	32
	402	1304	47.6636	37.52782	272.398	16	29	3	44
	432	1027	83.64937	36.64131	256.187	16	28	5	38
	420	1371	51.08242	36.98308	240.592	12	35	9	36
Non Touching	449	797	55.5688	34.03423	141.389	8	25	15	17
	489	1155	42.61806	36.27504	181.79	7	24	2	29
	285	847	46.49164	30.16958	156.663	26	5	2	28
	365	984	47.84766	31.93361	191.903	19	11	3	27
	348	885	50.96929	39.18898	173.741	23	12	2	30
	393	662	49.02589	29.05865	123.905	12	21	19	16

Table 1 Sample of feature set

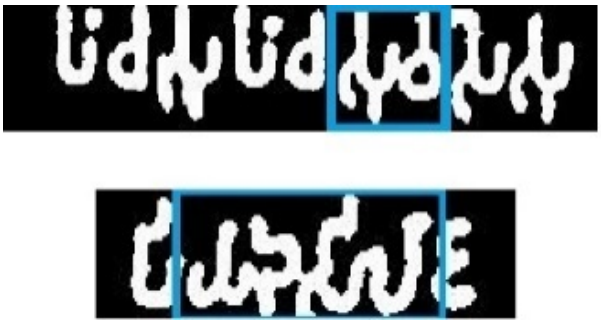

Fig. 3 Sample images of touching characters

We analyzed image set of 8014 (of which 963 are touching characters) segmented characters obtained from our previously developed algorithm for segmentation of Brahmi characters (Nagane and Mali (2020)).

4.2 Segmentation of touching characters

Once the touching characters are identified, they are segmented further into single characters. The touching characters may have two or more characters included. In our image set, the number of images containing two touching characters is the most. Three and four touching characters are also noticed considerably, whereas more number of touching characters are sparsely observed (Fig. 4). Table 2 describes this in detail. The pseudo-code for identification of the toughing characters is provided below:

Pseudo-code for identifying touching characters

```
for i =1 to n //n is number of images in a training directory
    read image;
    //generate feature vector FD containing different features
    TrainFD = [f1,f2,...,f9];
endfor;

for j =1 to m //m is number of images in a testing directory
    read image;
    //generate feature vector FD containing different features
    TestFD = [f1,f2,...,f9];
endfor;

train the model using TrainFD ensembled classification technique
test the trained model to predict the lables on TestFD
```

Number of touching characters	Frequency	Percentage (%)
2	743	9.27
3	172	2.14
4	41	0.51
5	6	0.07
6	NIL	0
7	1	0.01
8 or more	NIL	0
Total	963	12

Table 2 Details of touching characters

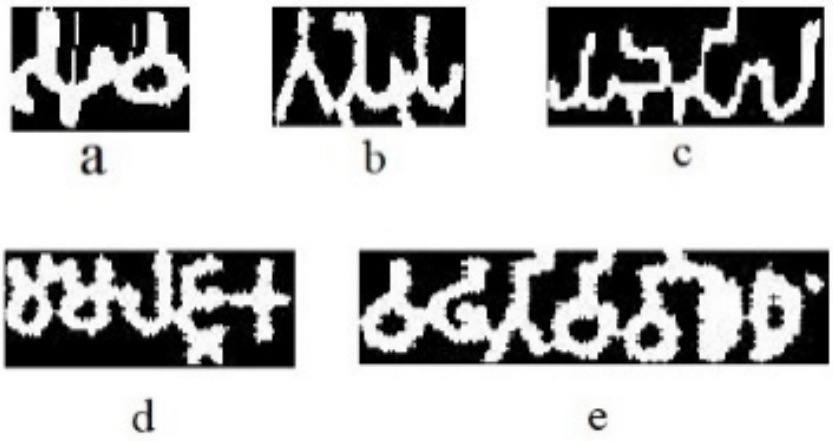

Fig. 4 Touching characters (a) two-touching characters; (b) three-touching characters; (c) four-touching characters; (d) five-touching characters; (e) seven-touching characters

To find correct segmentation points, initially we performed pre-processing on the image of touching characters. Morphological operations are executed in a sequence: closing, erosion, and thinning. For noise removal, median filter is applied after the 'close operation'. In the next step, we calculated the column-wise sum of foreground pixels. Further, we selected the columns as possible segmentation point (PSP) which has sum less than or equal to one. Then the final segmentation points (FSPs) were identified. For this, initially 7051 non-touching characters were analyzed to find the maximum number of columns required to represent the single Brahmi characters and a threshold was calculated.

Consider the first column as a marker (M). Compute the distance of column of PSP from M one by one until the distance is less than the threshold. Now, when the distance between the M and column from PSP is greater than threshold, add this column to FSP and also mark it as M. Repeat this process until all the columns from PSP are

checked. Now, FSP contains the correct segmentation points. So, using them we segment the touching characters successfully (Fig. 5). The pseudo-code for the segmentation of touching characters is given below:

Pseudo-code for segmenting touching character:

```
for i=1 to n // n is number of touching character images
        read image into variable img;
        // img is 2D matrix with 'r' rows and 'c' columns
        //perform preprocessing as below
        close operation with disk structuring element
        apply median filter to remove noise
        erode the image using same structuring element
        apply morphological operation "thin"
        //calculate each columns sum
        for i=1 to r
                colsum[i]=0;
                for j =1 to c
                        colsum[i] = colsum[i]+img[j][i];
                endfor
        endfor
        //Select Possible segmentation Points (PSP)
        PSP[1] = 1; // add first column of image in PSP
        j=2;
        for i=1 to c
                if colsum[i] <= 1
                        PSP[j++] = i; // Selecting PSP (column number)
                endfor
        endfor
        PSP[j] = w;// add last column of image in PSP
        //Select Final segmentation Points (FSP)
        Let M = PSP[1];
        j=1;
        for i = 2 to length(PSP)
                if(PSP[i]-M>threshold) //threshold is 20
                        FSP[j]=PSP[i]; // Selecting FSP (Column number)
                        M = PSP[i];
                endif
        endfor
        Segment or Crop the touching characters using FSP
endfor
```

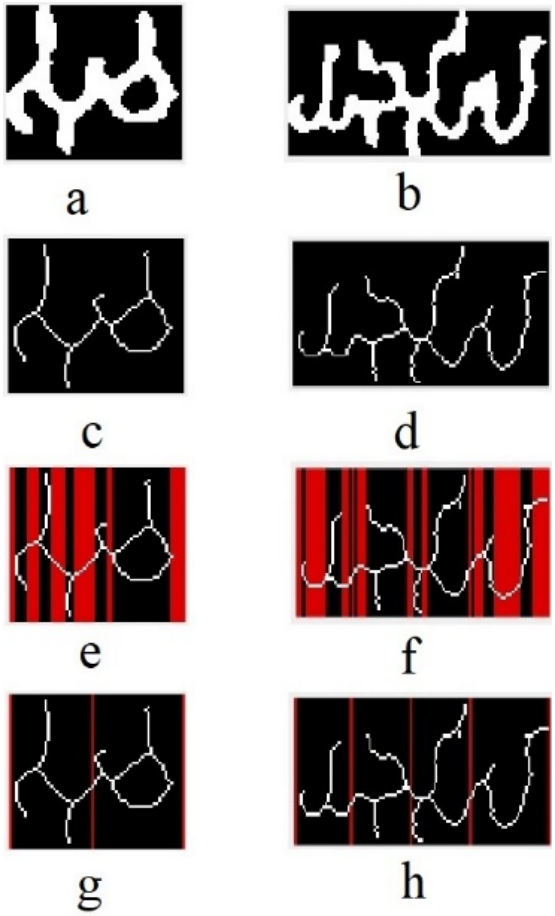

Fig. 5 Step-by-step segmentation of touching characters (a, b) touching characters; (c, d) images after the preprocessing; (e, f) selected PSPs; (g, h) selected FSPs

5 Results

The proposed method for identifying and segmentation of touching characters is implemented in two steps. Initially, we implemented the algorithm for identification of touching characters on the set of 8014 images (containing touching 963 touching characters). The algorithm identified 974 images of touching characters, out of which 963 images are identified correctly. Here, the recognition rate of touching characters was 100%. However, 11 extra single characters were identified as touching characters (Table 3). Furthermore, we implemented other classifiers to identify the touching characters which are support vector machine (SVM), K-nearest neighbour (KNN), and K-means.

Classification methods	Total number of images	Number of single characters	Number of touching characters	Number of touching characters identified correctly	Percentage of identification (%)
Ensembled	8014	7051	963	963	100
SVM	8014	7051	963	819	85
KNN	8014	7051	963	694	72
K-means	8014	7051	963	193	20

Table 3 Results of touching character identification

In the second phase, we implemented the algorithm for segmenting the touching characters on 963 images identified in previous step. These 963 images of touching characters contain 2203 individual characters. Total 955 images segmented successfully and resulted into correctly segmented 2187 individual characters. We achieved 99.16% of accuracy in segmentation of touching characters (Table 4).

Total number of images of touching characters	Number of individual characters in all images	Correctly segmented images	Correctly segmented individual characters	Percentage of identification (%)
963	2203	955	2187	99.16

Table 4 Results of touching character segmentation

Comparative analysis of various techniques used for the identification or segmentation or both is presented in Table 5. Methods compared are used on different scripts as no specific work in Brahmi script has been done so far.

Reference	Language/Script	Technique	Dataset Size	Result
Ullah et al. (2019)	Arabic	Overlapping Set Theory and Contour Tracing	220 word images	97.27%
Farulla and Murru (2017)	Latin	Fuzzy logic	1000 character images	Without Noise: 96.1% With Noise: 88.9% (Max):
Ali and Suresha (2019)	Arabic	Characters geometry and shape-based technique	13311 character images	92.5%
Garain and Chaudhuri (2002)	Devnagari and Bangla	Fuzzy Multifactorial Analysis	Devnagari: 11577 character images Bangla: 16714 character images	Devnagari: 57.48% Bangla: 53.65%
Deshmukh and Kolhe (2019)	Modi	Hybrid: Text zoning and column-wise background pixel density exploration	1812 character images	88.96%
Roy et al. (2009)	Latin	Dynamic programming using primitive segments	1790 character images	90.39%
Saba et al. (2011)	Latin	Neural Confidence: Geometric feature analysis and use of ANN	2936 character images	86.44%
Proposed Method	Brahmi	Connectivity based and sized based analysis	8014 character images	Identification: 100% Segmentation: 99.16%

Table 5 Comparison of results from various segmentation techniques.

6 Conclusion

In this paper, we have presented the method to identify and segment the touching characters from digitalized Brahmi script documents. The proposed method is implemented on already segmented images of characters using the earlier algorithm. The proposed methodology correctly identifies the touching characters and segments them into individual characters. The results are significant as we have achieved 100% accuracy in identification of touching characters using ensembled classification technique, whereas, the other techniques (SVM, KNN and K-means) have comparatively lower rate of touching character identification. Using the proposed method, we have achieved 99.16% of accuracy for segmentation of touching characters.
However, in the first phase, though we have identified all the toughing characters successfully, single characters were also identified as touching characters. Dynamic computation of threshold may help in addressing the problem. In case of segmentation of touching characters, there were characters that were not segmented accurately. This problem can be considered as over segmentation of the characters. So, there is a scope to work further on these problems and rectify them.

Acknowledgments

The authors would like to thank Dr. C. H. Patil for his academic support and valuable inputs in the research.

References

[1] Ali, A. A. A.; Suresha, M. (2019). An Efficient Character Segmentation Algorithm for Recognition of Arabic Handwritten Script. 2019 International Conference on Data Science and Communication (IconDSC), Bangalore, India, pp. 1–6, doi: 10.1109/IconDSC.2019.8817037.

[2] Burra, S.; Patel, A.; Bhagvati, C.; Negi, A. (2018). Improved Symbol Segmentation for TELUGU Optical Character Recognition. In: Abraham, A.; Muhuri, P.; Muda, A.; Gandhi, N. (Eds) *Intelligent Systems Design and Applications. ISDA 2017. Advances in Intelligent Systems and Computing*, vol. **736**. Springer, Cham. https://doi.org/10.1007/978-3-319-76348-4_48

[3] Deshmukh, M. S.; Kolhe, S. R. (2019). A Hybrid Character Segmentation Approach for Cursive Unconstrained Handwritten Historical Modi Script Documents (February 23, 2019). Proceedings of International Conference on Sustainable Computing in Science, Technology and Management (SUSCOM), Amity University Rajasthan, Jaipur - India, February 26-28, Available at SSRN: https://ssrn.com/abstract=3356213 or http://dx.doi.org/10.2139/ssrn.3356213

[4] Farulla, G. A.; Murru, N.; Rossini, R. (2017). A fuzzy approach to segment touching characters. *Expert Systems with Applications,* **88**, pp. 1–13.

[5] Garain, U.; Chaudhuri, B. B. (2002). Segmentation of touching characters in printed Devnagari and Bangla scripts using fuzzy multifactorial analysis. *IEEE Transactions on Systems, Man, Cybernet Ics Part C,* **32**, pp. 449–459. doi:10.1109/TSMCC.2002.807272.

[6] Garain, U.; Das, A.; Doermann, D. S.; Oard, D. D. (2012). Leveraging Statistical Transliteration for Dictionary-Based English-Bengali CLIR of OCR'd Text. Proceedings of COLING 2012: Posters, pp. 339-348.
[7] Hamad, K.A.; Kaya, M. (2016). A Detailed Analysis of Optical Character Recognition Technology. *International Journal of Applied Mathematics and Computer Science,* **4**(Special Issue), pp. 244-249.
[8] Jain, S. (2017). Optical Character Recognition System for Multilanguage Script Recognition. *International Journal of Recent Research Aspects,* **4**, pp. 153–159.
[9] Jha, V.; Parvathy, K. (2019). Braille Transliteration of Hindi Handwritten Texts Using Machine Learning for Character Recognition. *International Journal of Science and Technology Research*, **8**, pp. 1188–1193.
[10] Karthick, K.; Ravindrakumar, K.B.; Francis, R.; Ilankannan, S. (2019). Steps Involved in Text Recognition and Recent Research in OCR; A Study. *International Journal of Recent Technology and Engineering,* **8**, pp. 3095–3100.
[11] Liu, H.; Zhang, L. (2019). Advancing Ensemble Learning Performance through data transformation and classifiers fusion in granular computing context. *Expert Systems with Applications,* **131**, pp. 20–29. doi:10.1016/j.eswa.2019.04.051
[12] Mittal, R.; Garg, A. (2020). Text extraction using OCR: A Systematic Review. 2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA), Coimbatore, India pp. 357–362, doi: 10.1109/ICIRCA48905.2020.9183326.
[13] Nagane, A. S.; Mali, S. M. (2020). Segmentation of Characters from Degraded Brahmi Script Images. In: Iyer, B., Rajurkar, A., Gudivada, V. (Eds). *Applied Computer Vision and Image Processing. Advances in Intelligent Systems and Computing*, vol. **1155**. Springer, Singapore. https://doi.org/10.1007/978-981-15-4029-5_33
[14] Ojha, P. G. H. (1971). *Bharatiya Prachin Lipimala: The Palaeography of India*, Munshiram Manoharlal, New Delhi.
[15] Patil, C. H.; Mali, S. M. (2015). Segmentation of Isolated Handwritten Marathi Words. *IJCA Proceedings of National Conference on Digital Image Signal Processing DISP*. pp. 21–26.
[16] Peng, G.; Yu, P.; Li, H.; Li, H.; Zhu, X. (2017). A character segmentation algorithm for the palm leaf manuscripts, 2017 2nd IEEE International Conference on Computational Intelligence and Applications (ICCIA), Beijing, pp. 354–358, doi: 10.1109/CIAPP.2017.8167238.
[17] Rakshit, P.; Halder, C.; Ghosh, S.; Roy, K. (2018). Line, Word, and Character Segmentation from Bangla Handwritten Text—A Precursor Toward Bangla HOCR. In: Chaki, R.; Cortesi, A.; Saeed, K.; Chaki, N. (Eds). *Advanced Computing and Systems for Security. Advances in Intelligent Systems and Computing*, vol. **666**. Springer, Singapore. https://doi.org/10.1007/978-981-10-8180-4_7
[18] Ramalingam, K.; Bhojan, R. (2018). Identification of Broken Characters in Degraded Documents. *International Journal of Intelligent Engineering and Systems,* **11**, pp. 130–137.
[19] Rehman, A.; Saba, T. (2011). Performance analysis of character segmentation approach for cursive script recognition on benchmark database. *Digital Signal Processing*, **21**, pp. 486–490.
[20] Roy, P. P.; Pal, U.; Lladós, J.; Delalandre, M. (2009). Multi-Oriented and Multi-Sized Touching Character Segmentation Using Dynamic Programming. 2009 10th International Conference on Document Analysis and Recognition, Barcelona, pp. 11–15, doi: 10.1109/ICDAR.2009.124.
[21] Saba, T.; Rehman, A.; Elarbi-Boudihir, M. (2014). Methods and strategies on off-line cursive touched characters segmentation: a directional review. *Artificial Intelligence Review*, **42**, pp. 1047–1066. https://doi.org/10.1007/s10462-011-9271-5
[22] Saba, T.; Rehman, A.; Sulong, G. (2011). Cursive Script Segmentation with Neural Confidence. *International Journal of Innovative Computing Information and Control*, **7**, pp. 4955–4964.
[23] Salomon, R. (1998). *Indian Epigraphy: A Guide to the Study of Inscriptions in Sanskrit, Prakrit, and the Other Indo-Aryan Languages*. Oxford University Press, New York Oxford.
[24] Sen, S.; Chowdhury, S.; Mitra, M.; Schwenker, F.; Sarkar, R.; Roy, K. (2018). A novel segmentation technique for online handwritten Bangla words. *Pattern Recognition Letters* [in press]. https://doi.org/10.1016/j.patrec.2018.02.008
[25] Ullah, I.; Azmi, M. S.; Desa, M. I.; Alomari, Y. M. (2019). Segmentation of Touching Arabic Characters in Handwritten Documents by Overlapping Set Theory and Contour Tracing. *International Journal of Advanced Computer Science Applications*, **10**, pp. 155-160. URL: https://en.wikipedia.org/wiki/Estampage#:~:text=From%20Wikipedia%2C%20the%20free%20encyclopedia,paper%20for%20a%20clearer%20read [accessed on October 4, 2020]
[26] Zho, H.; Zhu, G.; Peng, Y. (2016). A RMB optical character recognition system using FPGA, 2016 IEEE International Conference on Signal and Image Processing (ICSIP), Beijing, pp. 539–542, doi: 10.1109/SIPROCESS.2016.7888320.

Authors Profile

Mr. Aniket Suresh Nagane is currently working as Assistant Professor at Department of Computer Science, MAEER's MIT Arts, Commerce and Science College, Alandi(D), Pune. He has completed his Masters' degree in Computer Science in 2009 and qualified NET examination. He is currently pursuing his Ph.D. from Savitriabai Phule Pune University, Pune under the guidance of Dr. S. M. Mali. He has published 5 research papers in national and international journals and has more than 12 years of experience in teaching.

Dr. Shankar Maruti Mali is currently working as Professor at School of Computer Science, MIT World Peace University. He is also the Coordinator of IQAC at MIT-WPU. He completed his PhD degree from Solapur University in 2011 and qualified SET in Computer Science in 2004. He is currently guiding 6 PhD scholars at Savitribai Phule Pune University and MIT World Peace University. His research areas include Digital image processing, Pattern recognition, Natural language processing and Networking. He has published 30 research papers in national and international journals. He has more than 22 years of experience in teaching. He is a member of Research and Recognition Committee at MIT-WPU and also a member of board of studies at different autonomous colleges in Pune. Previously, he has worked as Head of Department, Vice-Principal, BOS Chairman, Selection Committee Member and LIC Committee Member.