EAI Endorsed Transactions
on Creative Technologies Research Article **EAI.EU**

Non-Redundant Contour Directional Feature Vectors for Character Recognition

Tusar Kanti Mishra[1],*, Sandeep Panda[2], Banshidhar Majhi[3]

[1]Department of CSE, GITAM Deemed to be University, Visakhapatnam, India, 530045
[2]Goldman Sachs, Bengaluru, India
[3]Department of CSE, IIIT-DM, Kancheepuram, Chennai, India

Abstract

This paper presents a novel shape based feature for printed character recognition. The shape features are derived from the contour of the character which is unique to all characters. Preprocessing is performed to standardize the characters and handle all variations such as bold, italics and bold-italics font characteristics. The complete character set is clustered into different groups based on contour feature. A probe character is mapped into the corresponding cluster prior to recognition. This helps to reduce the computational overhead. Finally two recognition schemes have been proposed, based on angle feature extracted from the contour information and a longest common substring (LCS) based feature. Simulation has been carried out to validate the efficacy of the proposed scheme on printed *Odia* characters. Performance accuracy has been compared with the existing schemes. In general, it is observed that the proposed scheme outperforms the competitive schemes.

Received on 31 May 2020; accepted on 19 November 2020; published on 19 November 2020

Keywords: OCR, Odia character recognition, pattern recognition, classification, feature extraction.

doi:10.4108/eai.19-11-2020.167204

1. Introduction

One of the most important dialects of machine learning and computer vision is the task of Optical Character Recognition (OCR). OCR is the task where handwritten or printed set of characters are recognized by an electro-mechanical system and transformed into machine editable text. OCR has many real time applications that include automated document processing, signature verification, data entry from passports, invoices, bank documents, and digitization of printed texts so that they can be made available for machine processing. Many schemes have been implemented to handle the task of designing efficient OCR systems. In popular OCR systems, mostly characters are represented in a standard form to extract a feature set for each character followed by a classification of the characters using the features. Selection of features plays a vital role for achieving an efficient recognition measure. Selection of classifier is also equally important for optimal performance of OCR systems. Features used for OCR systems can be broadly classified into two major groups, i.e. shape based features and energy based features. Energy based features use some form of energy measures and shape based features try to capture the general shape detail of the characters. Some of the commonly used shape-based features are curvature-feature[1], f-ratio [2], zernike moments [3], and skeletal convexity [4]. In [5], a commercially viable *Telugu* character recognizer has been presented. It uses energy based features that include wavelet multi-resolution analysis and an associative memory models for feature extraction and recognition. For accomplishing the learning task Hopfield-based dynamic neural network is utilized. Printed *Odia* characters have been successively recognized using a combination of stroke and run-number based features [6]. Along with this, features are obtained using the concept of water overflow in a reservoir. However, no fine tuning of this work has been reported. Use of Hidden Markov Model (HMM) has been reportedly used in [7–10] for recognition of handwritten *Odia* numerals. The HMM is generated from the shape primitives of individual numeral and is referred to as template for establishing a match for probe numerals. For the matching task, its class conditional probability is found against each HMM. The Discrete Cosine Transform (DCT) and Discrete Wavelet Transform (DWT) based features algorithms for the purpose of handwritten

*Corresponding author. Email: tusar.k.mishra@gmail.com character recognition are studied in [11]. Resultant features separately from DCT and DWT have also been utilized in [12, 13] and comparative performance have been studied.

In [14], a character registration method has been presented for identification of movable type printing. This is based on the iterative Affine transformation applied to the contour of character images. For measuring the similarity among the shapes of characters, the bi-directional distance has been used. They have used coherent point drift algorithm for the purpose of character registration. For case study, *Tangut* characters are used.

In [15], modern feature extractor and classifier like convolutional neural networks (CNN) has been used for the purpose of cursive *Urdu* character recognition. LSTM (long short term memory unit [16]), a special variant of CNN has been utilized. In [17], a method based on recurrent neural networks (RNN) has been presented for *Chinese* character recognition. A combination of LSTM and GRU is utilized and validated on ICDAR-2013 dataset. The proposed framework is also capable of drawing pictorial *Chinese* characters. A fusion of both discriminative and generative approach is observed in the proposed framework. In [18], the authors have projected the problem of OCR as a image based sequencing problem. For this, they have suggested a trainable network for sequence term prediction. In [19], a framework namely ASTER has been presented that detects and recognize scene character images. Thine-plate spline along with sequence-to-sequence trainable model has been used for the purpose.

ଅ ଆ ଇ ଈ ଉ ଊ ଋ ଏ ଐ ଓ

ଔ କ ଖ ଗ ଘ ଚ ଛ ଜ ଝ ଞ

ଟ ଠ ଡ ଢ ଣ ତ ଥ ଦ ଧ ନ

ପ ଫ ବ ଭ ମ ଯ ର ଲ ଳ ସ

ଶ ଷ ହ କ୍ଷ

Figure 1. *Odia* characters set.

2. Proposed Scheme

Robust and conventional heuristics are still a demand for efficient OCR problems. Existing OCR systems are generally utilizing deep neural network concepts. This may not be always useful completely for symbols of every languages. *Odia* is one of the official languages of India. It has recently been honored with the status of classical language in the Indian subcontinent. The character set used in the *Odia* language is shown in Fig.1. As each language comes with its unique characters, stroke styles and writing schemes, hence developing a single scheme for all scripts is very difficult. Thus, for common languages, algorithms are proposed which are capable of uniquely recognizing the scripts under consideration. In this paper, we propose a shape based feature considering the shape delicacy, stroke styles of each character in *Odia* language. Further a scheme is proposed to handle the italics, bold faced, and characters with different font types prior to feature extraction. Rest of the paper is organized as follows. Section 2 describes the proposed scheme in detail. Section 3 deals with the experimental results and performance analysis. Finally, Section 5 provides the concluding remarks and prospects for future work. The scheme considers an input string of *Odia* characters and attempts to recognize each character in the string. It emphasizes on preprocessing, feature extraction, and recognition. Overall steps followed in the proposed scheme are given in Fig. 2. Different steps performed in the scheme are discussed in the subsequent sections in sequel.

2.1. Preprocessing

Preprocessing is the first step in any character recognition system. The prime importance of this step lies in the fact that characters must be standardized before any resemblance remark can be made. This essentially means that all characters must be in the same state prior to comparison. Here, state of a character refers to the font size, italics, and bold nature or any such characteristics which may vary from character to character. Correction of such variations and normalizing the characters is an essential requirement in character recognition. Further, extraction of the atomic characters from a composite string of characters is of utmost importance. An overall block diagram of the preprocessing steps involved in our scheme is shown in Fig. 3. The preprocessing is essentially carried out in four major steps in the proposed work as discussed below.

Character Atomization. In any character recognition scheme, e.g. license plate recognition, bank document recognition, invoice classification etc., are normally supplied as a string of characters (Fig. 4). Prior to recognition of the characters, it is needed to separate the individual characters.

To extract the individual characters, we utilize a vertical histogram of the entire string image. The image is traversed from left to right. At each column position a vertical scan line is constructed and the number of foreground pixels are recorded. The histogram is constructed depending on the frequency of the foreground pixel against the position of the scan line. The histogram obtained for the input character string is shown in Fig. 5.

It may be observed that, the histogram has zero frequencies of foreground pixels at certain intervals which denote the spaces between the characters. The midpoints of these zero frequencies are extracted to separate two different characters. Each character is subsequently represented as a image size of 64×64.

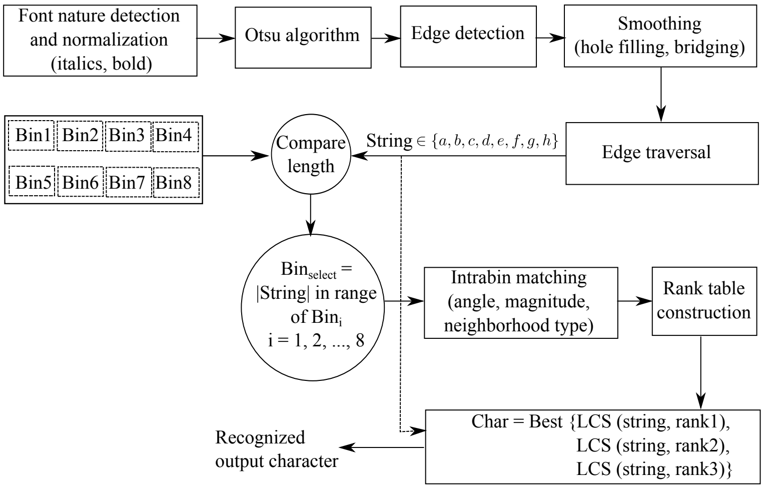

Font nature detection and normalization (italics, bold)
Otsu algorithm
Edge detection
Smoothing (hole filling, bridging)
Edge traversal
String $\in \{a,b,c,d,e,f,g,h\}$
Bin1 Bin2 Bin3 Bin4
Bin5 Bin6 Bin7 Bin8
Compare length
$\text{Bin}_{\text{select}}$ = |String| in range of Bin_i i = 1, 2, ..., 8
Intrabin matching (angle, magnitude, neighborhood type)
Rank table construction
Char = Best {LCS (string, rank1), LCS (string, rank2), LCS (string, rank3)}
Recognized output character

Figure 2. Overall block diagram of the proposed scheme.

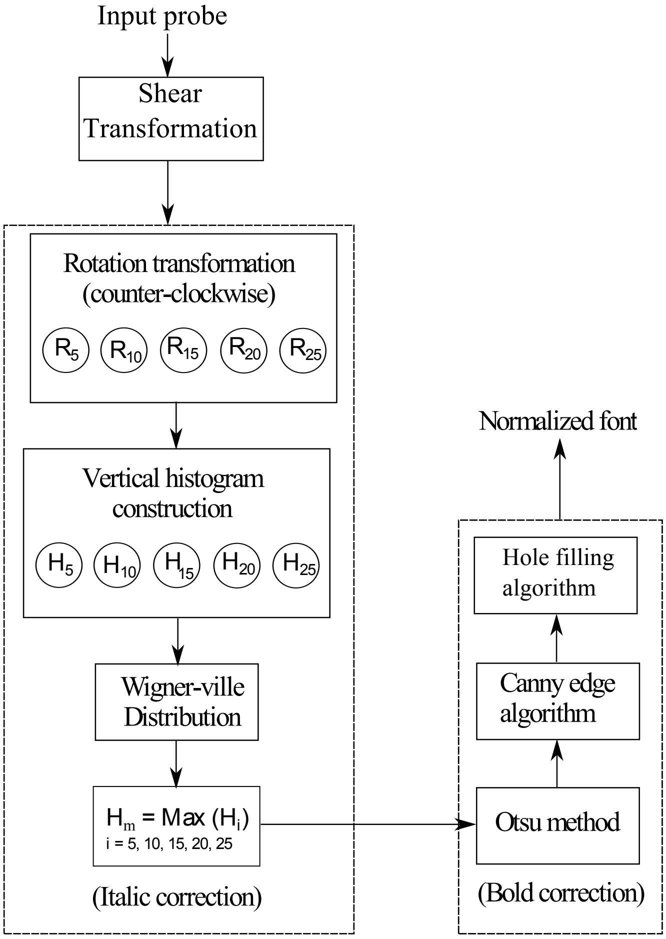

Input probe
Shear Transformation
Rotation transformation (counter-clockwise)
R_5 R_{10} R_{15} R_{20} R_{25}
Vertical histogram construction
H_5 H_{10} H_{15} H_{20} H_{25}
Wigner-ville Distribution
H_m = Max (H_i) i = 5, 10, 15, 20, 25
(Italic correction)
Normalized font
Hole filling algorithm
Canny edge algorithm
Otsu method
(Bold correction)

Figure 3. Overall block diagram of the preprocessing steps.

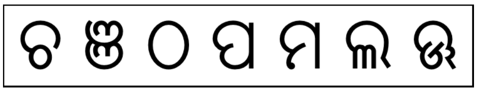


Figure 4. A sample input string of *Odia* characters.

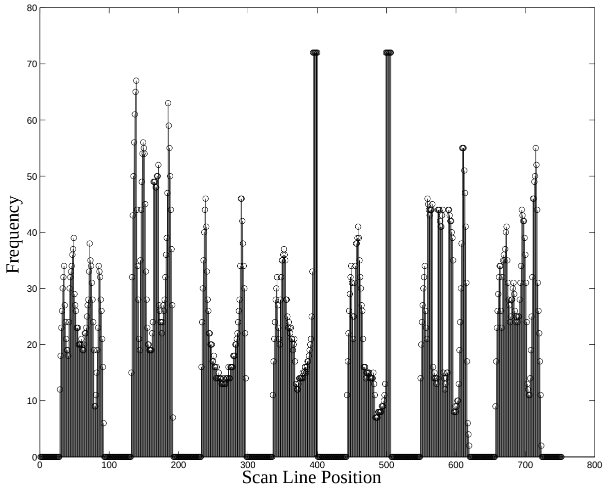


Figure 5. Histogram of the input sequence.

Font Normalization. Normally, the printed characters can appear either in normal, italics, bold or bold-italics depending upon the emphasis put on the words of a sentence. If we extract features after character atomization it may lead to serious error during recognition. Hence, the task of font normalization plays an important role to standardize all the images prior to feature extraction. The major corrections involved in this step are the italics correction and bold character rectification. In our scheme, the italics and bold corrections are performed in sequence. The order is influenced by the scheme implemented to rectify these variations. Italics is essentially the tilt introduced in printed characters to provide emphasis on the character. A sample of a *Odia* character in italics is shown in Fig. 6.

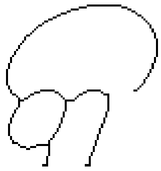


Figure 6. Italicized *Odia* character sample.

To perform italics correction, the image is initially subjected to shear and rotation transform. The combined transform matrix is given by,

$$\begin{pmatrix} cos\theta + \lambda sin\theta & -sin\theta + \lambda cos\theta \\ sin\theta & cos\theta \end{pmatrix} \qquad (1)$$

where, λ is the shear parameter and
θ is the rotation angle in anticlockwise direction

The amount of shear introduced is fixed for all italics and experimentally the value of λ is found to be 0.3. However, the amount of rotation required is not fixed for the characters. The normal form is determined by the use of Wigner-Ville Distribution(WVD) [20]. A vertical histogram is constructed for the transformed character and is used as a feature for the WVD. The WVD provides the spectral energy density of the characters and thus the normal character is identified as the one that has the highest energy density in the WVD.

The rotation angle θ is varied from 0^0 to 30^0 in a step of 5^0 and each time the histogram is plotted to identify the spectral energy in each case. It is observed that at $\theta = 20^0$, the spectral energy density is maximum which represents the normal font of the character. A sample rotated *Odia* 'ka' is shown in Fig. 7. The transformed image is subjected to hole filling to remove the discontinuities obtained due to the rotation.

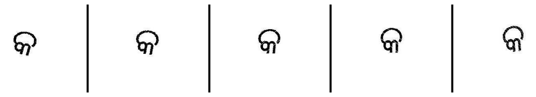


Figure 7. From left to right: Images rotated by 5^0 in anticlockwise direction in sequence.

Once the italics correction is complete, the image is processed for bold correction. Characters are usually made bold to signify special meaning or importance to a word. It can be noted that any general text editor allows one bold effect. This is to say that a normal can be made bold only once in printed text. To perform bold correction, the probe image is thinned once and thickened once. This will ensure that normal character is obtained as one of the three images obtained in this process. This is shown in Fig. 8. As can be observed, the normal variation of a bold character is obtained.

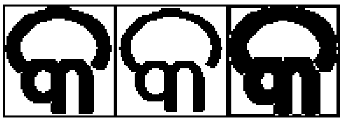


Figure 8. From left to right: Bold character, One level thinned , One level thickened.

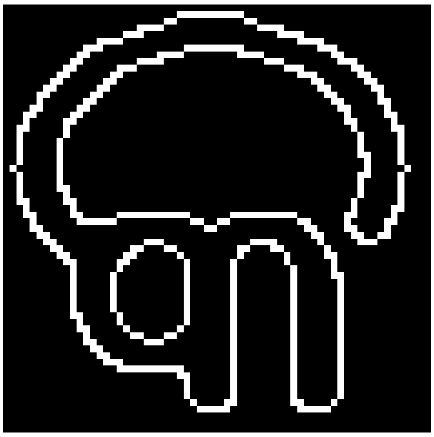
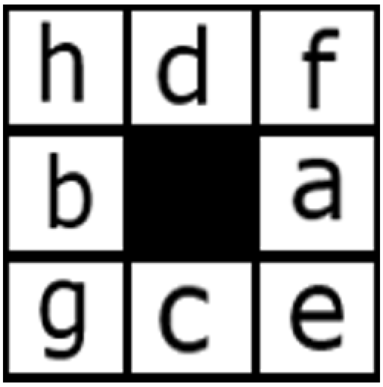


Figure 9. Final preprocessed image contour.



h	d	f
b		a
g	c	e

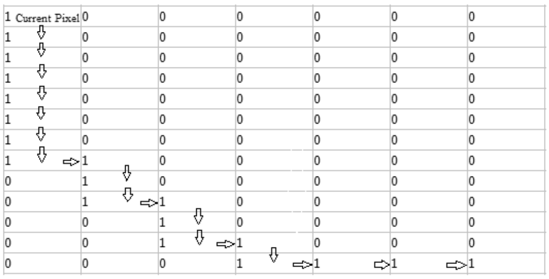
Figure 10. Demonstrating the traversal matrix.

Contour formation. The final step in the preprocessing involves preparing the standardized character for the proposed algorithm. In this step, the image is first thresholded using the Otsu's transform. The Otsu thresholding method is applied to reduce the gray scale image into corresponding binary equivalent image. In the next step, binarization is carried out. After this step, the contour of the image is constructed from the binarized image. Finally, morphological bridging is carried out to fill any holes on the contour of the character. Thus the image is reduced to a single $m-connected$ component along the contour. A sample of a final preprocessed image that is taken as an input to the algorithm is shown in Fig. 9.

2.2. Feature Extraction

Feature extraction is the most important phase of any character recognition scheme. Features give a unique representation of a character which is utilized to identify the characters. In this work we have proposed a new contour based shape feature. The image is originally scanned from left to right until the first foreground pixel is found. From this point, the image is traversed in a clockwise direction in search of the next foreground pixel. For every foreground pixel a 3×3 neighborhood is constructed with different directions a,b ... g, and h as shown in Fig. 10. Direction of the next foreground pixel is first searched in $4-neighborhood$ and if not found, the search is extended to $8-neighbourhood$. Once we obtain the next foreground pixel, its direction of search is recorded and the search for the next foreground pixel is continued from the newly observed pixel. This process is repeated for all the pixels, i.e. the search is completed when we reach the first pixel that we have started. This process results in a search string of directions i.e. a contour direction vector (CDV) which is unique for each class of character in terms of length and order of directions.

.



1 Current Pixel	0	0	0	0	0	0
1 ⇩	0	0	0	0	0	0
1 ⇩	0	0	0	0	0	0
1 ⇩	0	0	0	0	0	0
1 ⇩	0	0	0	0	0	0
1 ⇩	0	0	0	0	0	0
1 ⇩	0	0	0	0	0	0
1 ⇩	⇨1	0	0	0	0	0
0	1 ⇩	0	0	0	0	0
0	1 ⇩	⇨1	0	0	0	0
0	0	1 ⇩	0	0	0	0
0	0	1 ⇩	⇨1	0	0	0
0	0	0	1 ⇩	⇨1	⇨1	⇨1

(a) A section of the contour showing the traversal. The arrows indicate the direction of the traversal in the section.

ccccccccaccaccaca

(b) The corresponding string generated from the above image section.

cacacaca

(c) The corresponding non-redundant string.

Figure 11. The final string generation.

Repetitions of directions in the CDV signifies that the contour is in the same direction for the number of times the direction is repeated. These repetitions will vary as the size of the character is different and hence CDV is not scale invariant. To achieve scale invariance, the redundant directions are merged into one direction and a non-redundant contour direction vector (NRCDV) is constructed which reflects the unique change of contour directions in a particular character of any font size. Here, the contour traversal (Fig. 11(a)) results in the CDV (Fig. 11(b)) and it again in turn result in the NRCDV(Fig. 11(c)). As the stroke and angle of each character is unique in NRCDV, a set of angle features from NRCDV namely, angle magnitude (AM), angle rotation (AR), and neighborhood belongingness (NB) are computed as follows,

ପ
ଘଠଷସ
ଯ
ଟବ
ଏଋଌଚଛଜଝଦଭ
ଇଈକଖଗଙଢଣତଧନହଶମୟଳର
ଐଡଥଲଅ
ଆଉଊଔଞ

Figure 12. Clusters by string lengths.

1. Angle magnitude(AM):

$$\begin{cases} 0, & \text{if the next pixel is along the same row} \\ 45, & \text{if the next pixel is along the diagonal and the diagonal direction} \\ 90, & \text{if the next pixel is along the same column} \\ 135, & \text{if the next pixel is along the diagonal and opposite it's direction} \end{cases}$$

2. Angle of rotation(AR):

$$\begin{cases} 0, & \text{if next pixel is clockwise when rotated along exterior of traversal matrix} \\ 1, & \text{if next pixel is anti-clockwise when rotated along exterior of traversal matrix} \end{cases}$$

3. Neighborhood belongingness(NB):

$$\begin{cases} 1, & \text{if current pixel is a 4-neighbour and next pixel is a diagonal neighbour} \\ 2, & \text{if current pixel is a diagonal neighbour and next pixel is a 4-neighbour} \\ 3, & \text{if current pixel is a 4-neighbour and next pixel is also a 4-neighbour} \end{cases}$$

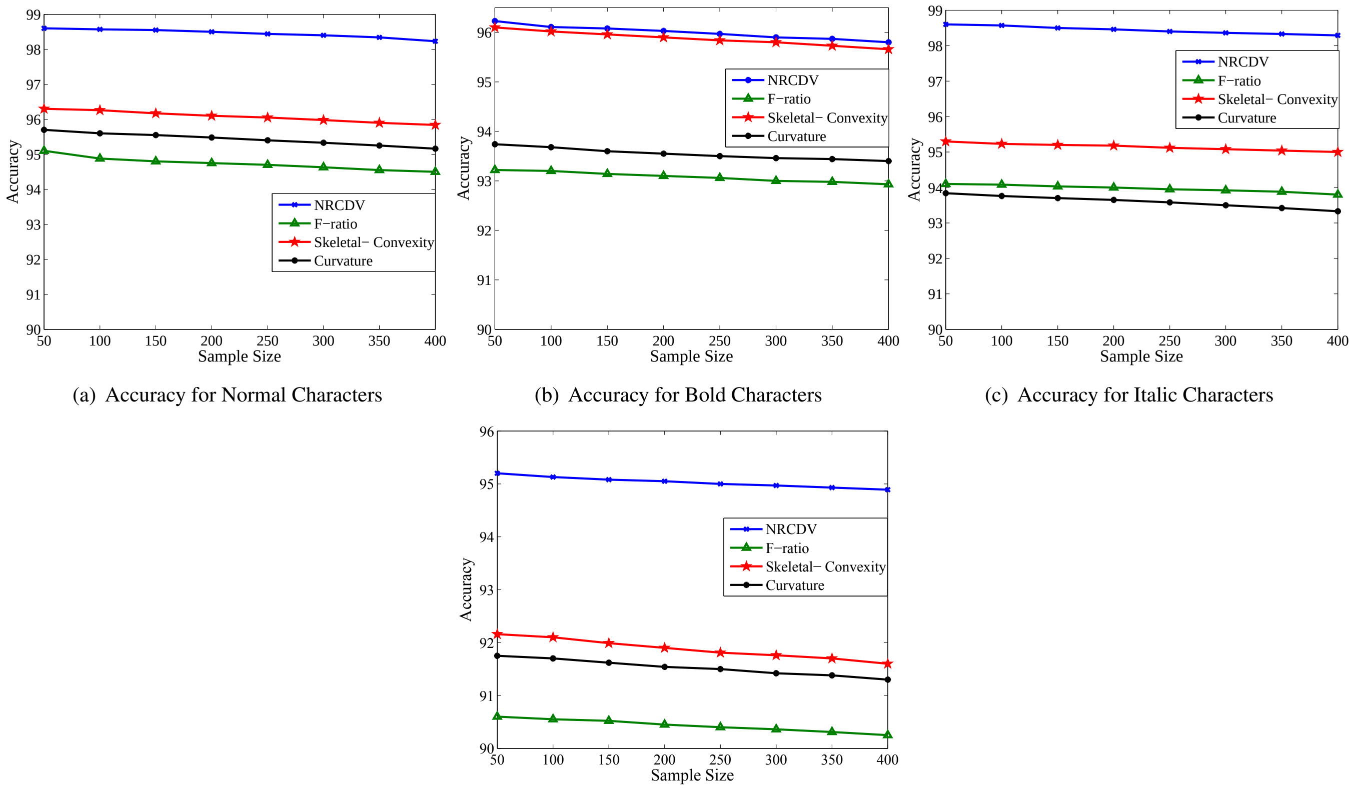
Thus it may be noted that so far we have captured the followings for a particular character,

1. Contour direction vector (CDV)
2. Non-redundant contour direction vector (NRCDV)
3. Angle magnitude vector (AMV)
4. Angle rotation vector (ARV)
5. Neighborhood belongingness vector (NBV)

All the five vectors are used as features for a specific character which has undergone italics and bold corrections. Subsequently these are used for recognition. The term non-redundant conveys that the feature points have been uniquely pooled so that there will not be redundant values in any of the subsequence in the final feature vector.

2.3. Character Recognition

It may be noted that *Odia* language consists of 47 characters out of which 12 of those are considered as *swara varna* and rests are called as *vyanjana varna*. To expedite recognition



(a) Accuracy for Normal Characters

(b) Accuracy for Bold Characters

(c) Accuracy for Italic Characters

(d) Accuracy for Bold-Italic Characters

Figure 13. Accuracy Comparison with other competent schemes.

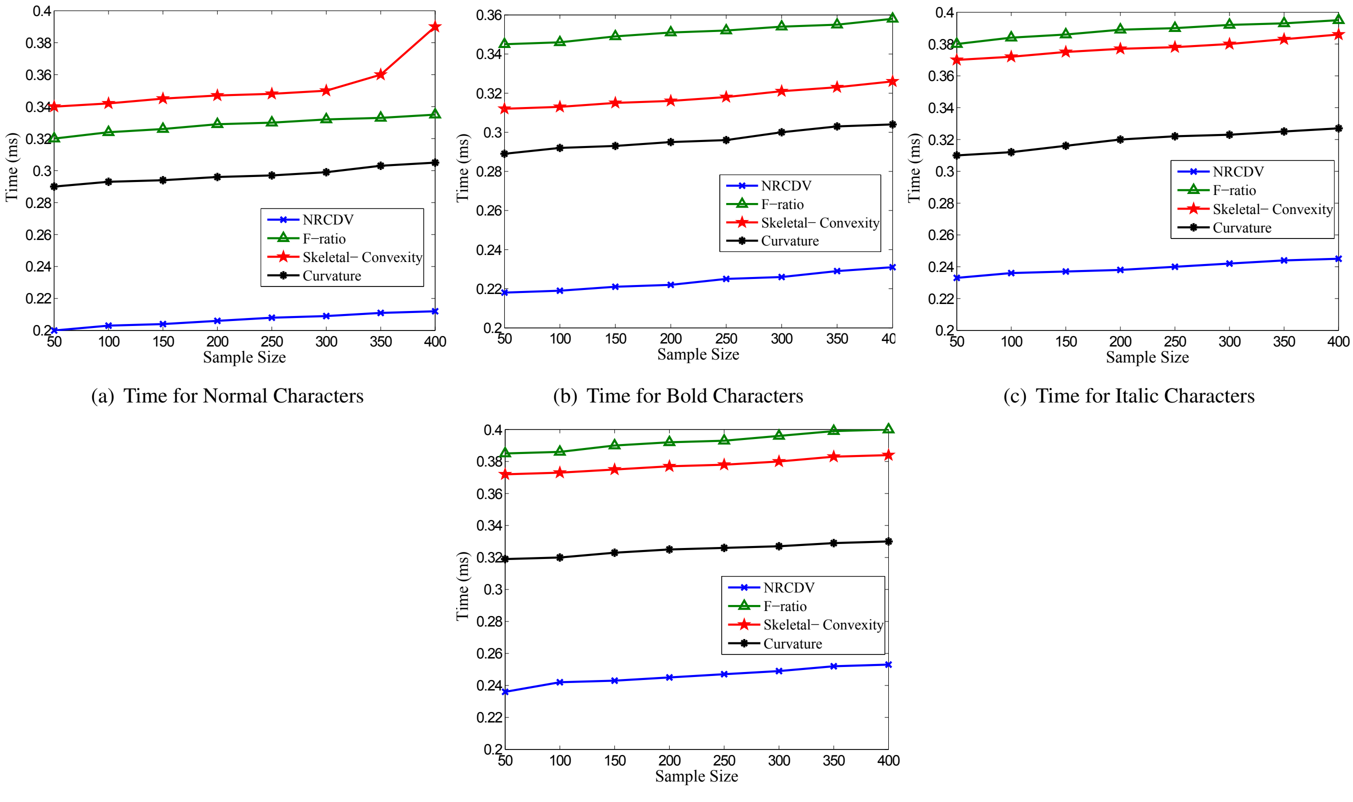
of any character, the whole character set is partitioned into several clusters based on the length of CDV, where each cluster length is in a range such as $0-50, 51-100$, etc. Let the clusters be $c_1, c_2, \ldots, c_m$ with $p_1, p_2, \ldots, p_n$ CDV string lengths respectively. The typical clustering of *Odia* character set is shown in Fig. 12 with $m = 8$. At the time of recognition, the probe character is hashed into its corresponding cluster and subsequent recognition operations are performed with the characters in the cluster in place of all the characters and there by it leads to reduction of computational cost significantly.

If the number of characters in a cluster exceeds 4, to find the top best matches, angle features i.e. AMV, ARV, and NBV are compared using k-nearest neighbor (KNN) algorithm which uses majority voting [21]. Euclidean distance measure is considered for this purpose. In the final step, longest common subsequence (LCS) algorithm ([22–24]) is used in the top order ranked characters with respect to NRCDV and the character with maximum matched subsequence is considered to be the target class.

3. Experimental results and analysis

To validate the efficacy of the proposed scheme, simulation has been carried out on *Odia* printed characters. *Odia* characters of several fonts such as Sarala, Akshara, and Kalinga are considered along with normal, bold, italics, and bold-italics styles. Each character in the dataset is standardized to 64×64 image. Subsequently each character image is subjected to preprocessing for bold, italics, and bold-italics corrections followed by contour detection. The features CDV, NRCDV, AMV, ARV, and NBV are extracted for each character. The whole dataset is clustered using CDV with respect to range of lengths as $0-50, 51-100, \ldots, 300-350$ and only non-empty clusters are considered.

The probe character is subjected to similar preprocessing and it is initially mapped to a cluster. If the cluster length is greater than 3, ranking is performed using angle features (AMV, ARV, NBV) using KNN classification. From a set of 350 probe characters, the rank identification table is given in Table 1 in a percentage measure. It may be observed that, most of the characters have rank-1 accuracy and rest of them are with in rank-3 accuracy. The accuracy, penetration rate, bin hit rate, and computational overhead for various characters chosen from various fonts with normal, bold, italics, and bold-italics are listed in Table 2. It may be observed that an overall recognition accuracy rate of 95% is obtained through the k-fold (k=5) cross-validation method. The comparison of overall rate of accuracy is shown in Fig. 13 shows an improved performance when compared to existing schemes. Further, a time comparison for recognition of various sample sizes with different fonts are shown in Fig. 14.



(a) Time for Normal Characters

(b) Time for Bold Characters

(c) Time for Italic Characters

(d) Time for Bold-Italic Characters

Figure 14. Comparison of computation overhead among various schemes.

4. Discussion

Table 1. Rank-wise rates of classification accuracy.

Font	Rank-1	Rank-2	Rank-3
Normal	97%	2%	1%
Bold	96%	2%	2%
Italics	94%	2%	4%
Bold – italics	93%	2%	5%

The proposed NRCDV feature vector is unique in nature. It is scale invariant so that the size of an input character wont effect the recognition performance. Time of computation is also efficient enough once the vectors are generated. Due to removal of redundant values in the feature vector, the pre-storage requirement have been drastically reduced. Thus, effective implementation of the scheme for lower level hardware devices is also possible. These The proposed scheme has lowest computational overhead. The research work has been well validated on the perfect characters. Yes, the applications are vivid. There are newspapers published in this language with daily circulations of approximately 1 million. The suggested work can be proper utilized in this scenario. Many people are illiterate in the region. A proper application can be suitably designed with the proposed theme which can facilitate an easy third party automated reading of the newspapers for those illiterates.

Table 2. Experimental results for various *Odia* fonts (N: Normal; I: Italics; B: Bold; BI: Bold Italics).

Font	Font Type	Rate of Penetration	Bin Hit Rate	Overall Accuracy	Time for individual character	Average Match Percentage of Strings
Kalinga	N	100	97.87	97.87	0.21	92.65
Kalinga	I	100	95.74	95.74	0.24	90.68
Kalinga	B	100	97.87	97.87	0.223	91.86
Kalinga	BI	100	95.74	95.74	0.264	88.67
Aprant	N	100	97.87	97.87	0.21	92.65
Aprant	I	97.87	95.74	93.61	0.238	91.3
Aprant	B	100	97.87	97.87	0.216	91.81
Aprant	BI	97.87	95.74	93.61	0.247	90.58
Sarala	N	97.87	100	97.87	0.207	93.15
Sarala	I	97.87	97.87	95.74	0.236	91.67
Sarala	B	97.87	100	97.87	0.216	92.54
Sarala	BI	97.87	95.74	93.61	0.246	90.82
Khandadhar	N	100	100	100	0.197	96.54
Khandadhar	I	100	97.87	97.87	0.223	93.76
Khandadhar	B	100	100	100	0.214	95.84
Khandadhar	BI	100	97.87	97.87	0.229	94.84
Brahmani	N	100	100	100	0.206	95.23
Brahmani	I	100	97.87	97.87	0.23	92.96
Brahmani	B	100	100	100	0.221	94.11
Brahmani	BI	100	95.74	95.74	0.246	92.34
TaraTarini	N	97.87	97.87	95.74	0.226	92.35
TaraTarini	I	97.87	95.74	93.61	0.25	90.66
TaraTarini	B	97.87	97.87	95.74	0.242	91.12
TaraTarini	BI	97.87	95.74	93.61	0.233	88.94
Mahanadi	N	100	100	100	0.207	95.84
Mahanadi	I	100	97.87	97.87	0.25	93.16
Mahanadi	B	100	100	100	0.22	93.54
Mahanadi	BI	100	95.74	95.74	0.239	91.35
Aksar	N	100	97.87	97.87	0.206	93.2
Aksar	I	100	95.74	95.74	0.23	90.61
Aksar	B	100	97.87	97.87	0.218	91.22
Aksar	BI	97.87	95.74	93.61	0.257	89.63

5. Conclusion

In this paper we have suggested a robust feature detection scheme for recognition of printed characters. The scheme has been generalized for multiple font types. Experimental results show satisfactory recognition rates for four different characteristics of *Odia* fonts. The computational overhead of the proposed scheme has also been found to be far better than other competent character recognition schemes. However, some improvements in the time complexity can also be brought by looking into some more data structures that can speed up the matching process along with more efficient storage algorithms. Efficient storage algorithm would be beneficial towards faster computation that may lead to obtain approximately real time output. Further, the scheme can be extended for handwritten character recognition.

References

[1] U. Pal, T. Wakabayashi, and F. Kimura, “A system for off-line oriya handwritten character recognition using curvature feature.” in *ICIT*. IEEE Computer Society, 2007, pp. 227–229.

[2] T. Wakabayashi, U. Pal, F. Kimura, and Y. Miyake, “F-ratio based weighted feature extraction for similar shape character recognition,” in *ICDAR*, 2009, pp. 196–200.

[3] A. Khotanzad and Y. Hong, “Invariant image recognition by zernike moments,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, pp. 489–497, 1990.

[4] S. Bag, P. Bhowmick, and G. Harit, “Recognition of Bengali handwritten characters using skeletal convexity and dynamic programming,” *Emerging Applications of Information Technology, International Conference on*, pp. 265–268, 2011.

[5] A. K. Pujari, C. D. Naidu, M. S. Rao, and B. Jinaga, “An intelligent character recognizer for telugu scripts using multiresolution analysis and associative memory,” *Image and Vision Computing*, vol. 22, no. 14, pp. 1221 – 1227, 2004.

[6] B. Chaudhuri, U. Pal, and M. Mitra, “Automatic recognition of printed oriya script,” *Sadhana*, vol. 27, pp. 23–34, 2002.

[7] T. K. Bhowmik, S. K. Parui, U. Bhattacharya, and B. Shaw, “An hmm based recognition scheme for handwritten oriya numerals,” in *Proceedings of the 9th International Conference on Information Technology*, ser. ICIT ’06. Washington, DC, USA: IEEE Computer Society, 2006, pp. 105–110.

[8] T. K. Mishra, B. Majhi, P. K. Sa, and S. Panda, “Model based odia numeral recognition using fuzzy aggregated features,” *Frontiers of Computer Science*, vol. 8, no. 6, pp. 916–922, Dec 2014.

[9] J. Hu, M. K. Brown, and W. Turin, “Hmm based online handwriting recognition,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 10, pp. 1039–1045, Oct 1996.

[10] “Hidden markov model based optical character recognition in the presence of deterministic transformations,” *Pattern Recognition*, vol. 26, no. 12, pp. 1813 – 1826, 1993.

[11] T. Mishra and A. Pujari, “Time series quanslet: A novel primitive for image classification,” in *Contemporary Computing*, ser. Communications in Computer and Information Science. Springer Berlin Heidelberg, 2012, vol. 306, pp. 64–72.

[12] I. Batal and M. Hauskrecht, “A supervised time series feature extraction technique using dct and dwt,” in *Machine Learning and Applications, 2009. ICMLA ’09. International Conference on*, 2009, pp. 735–739.

[13] M. Dale and M. Joshi, “Fingerprint matching using transform features,” in *TENCON 2008 - 2008 IEEE Region 10 Conference*, 2008, pp. 1–5.

[14] F. Sun, Y. Lin, Z. Zhang, D. Qu, and X. Li, “Movable type printing identification based on tangut characters registration,” *Neurocomputing*, vol. 252, pp. 87 – 94, 2017.

[15] S. Naz, A. I. Umar, R. Ahmad, I. Siddiqi, S. B. Ahmed, M. I. Razzak, and F. Shafait, “Urdu nastaliq recognition using convolutionalâĂŞrecursive deep learning,” *Neurocomputing*, vol. 243, pp. 80 – 87, 2017.

[16] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Computation*, vol. 9, pp. 1735 – 1780, 1997.

[17] X. Zhang, F. Yin, Y. Zhang, C. Liu, and Y. Bengio, “Drawing and recognizing chinese characters with recurrent neural network,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 4, pp. 849–862, April 2018.

[18] B. Shi, X. Bai, and C. Yao, “An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 11, pp. 2298–2304, Nov 2017.

[19] B. Shi, M. Yang, X. Wang, P. Lyu, C. Yao, and X. Bai, “Aster: An attentional scene text recognizer with flexible rectification,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2018.

[20] N. Bergmann, “New formulation of discrete wigner-ville distribution,” *Electronics Letters*, vol. 27, no. 2, pp. 111–112, Jan 1991.

[21] T. Cover and P. Hart, “Nearest neighbor pattern classification,” *Information Theory, IEEE Transactions on*, vol. 13, no. 1, pp. 21–27, January 1967.

[22] D. Maier, “The complexity of some problems on subsequences and supersequences,” *J. ACM*, vol. 25, no. 2, pp. 322–336, Apr. 1978.

[23] F. Nicolas and E. Rivals, “Longest common subsequence problem for unoriented and cyclic strings,” *Theoretical Computer Science*, vol. 370, no. 1âĂŞ3, pp. 1 – 18, 2007.

[24] L. Bergroth, H. Hakonen, and T. Raita, “A survey of longest common subsequence algorithms,” in *String Processing and Information Retrieval, 2000. SPIRE 2000. Proceedings. Seventh International Symposium on*, 2000, pp. 39–48.

Dr. Tusar Kanti Mishra is having expertise in the field of optical character recognition. Presently, he is working as an associate professor in the department of computer science and engineering in Gayatri Vidya Parishad College of Engineering for Women at Visakhapatnam, India. He has earned his full-time doctorate degree in the field of pattern recognition from National Institute of Technology, Rourkela, India. He is also an Erasmus Fellow for which he has worked as a visiting research scientist in The RUG, The Netherlands for a period of six months with full fellowship. He He has a strong background of research and teaching work since 2004. His areas of research interest include pattern recognition, image processing, computer vision and artificial intelligence.

Mr. Sandeep Panda iis a dynamic personality in the field of programming. He is serving the popular software company Goldman Sachs at Bengaluru, India. He has received multiple numbers of internships in his early career. He has earned his B.Tech degree in computer science and engineering from National Institute of Technology, Rourkela, India. In his early stage of career, he has been successfully published numerous quality articles in journals and conferences of repute.

Prof.(Dr.) Banshidhar Majhi is an eminent professor and a veteran researcher in the field of pattern recognition and image processing. He is the Director of IIIT-DM, Kancheepuram, Chennai, India. He has published more than hundred of quality articles in referred and indexed journals. He has also worked as the Dean(AA) in NIT Rourkela. Prof. Majhi has successfully guided thirty-five doctorates till date. He has research expertise in the domain of pattern recognition, image processing, artificial intelligence, and wireless sensor networks.