IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 12, 2017, accepted January 11, 2018, date of publication January 18, 2018, date of current version March 15, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2795104

Multilingual Character Segmentation and Recognition Schemes for Indian Document Images

PARUL SAHARE AND SANJAY B. DHOK

Centre for VLSI & Nanotechnology, Department of Electronics and Communication Engineering, Visvesvaraya National Institute of Technology, Nagpur 440010, India

Corresponding author: Parul Sahare (parulsahare2387@gmail.com)

ABSTRACT In this paper, robust algorithms for character segmentation and recognition are presented for multilingual Indian document images of Latin and Devanagari scripts. These documents generally suffer from their layout organizations, local skews, and low print quality and contain intermixed texts (machine-printed and handwritten). In the proposed character segmentation algorithm, primary segmentation paths are obtained using structural property of characters, whereas overlapped and joined characters are separated using graph distance theory. Finally, segmentation results are validated using highly accurate support vector machine classifier. For the proposed character recognition algorithm, three new geometrical shape-based features are computed. First and second features are formed with respect to the center pixel of character, whereas neighborhood information of text pixels is used for the calculation of third feature. For recognizing the input character, k-Nearest Neighbor classifier is used, as it has intrinsically zero training time. Comprehensive experiments are carried out on different databases containing printed as well as handwritten texts. Benchmarking results illustrate that proposed algorithms have better performances compared to other contemporary approaches, where highest segmentation and recognition rates of 98.86% and 99.84%, respectively, are obtained.

INDEX TERMS Character recognition, character segmentation, document analysis, graph theory, multilingual Indian optical character recognition.

I. INTRODUCTION

Automated Optical Character Recognition (OCR) finds an important application in building the intelligent indexing systems in today's computer world. This recognition process digitizes the documents and converts them into editable forms. Document image analysis is an area, which deals with scanned images. Apart from this indexing system, other industrial applications of OCR are found in drug and food industries to keep track of life cycle of goods. OCR performance for industries is mainly dependent on the quality of camera, whereas for document analysis, it depends upon the scanner. Non-uniform illumination within the scanner due to shadow, scanning angle and ink diffusion are some of the main problems encountered while scanning.

OCR usually involves three processes, namely text localization, character segmentation and recognition. The recognition is an important area of document image analysis, which is in mature stage for machine-printed text. However, for intermixed texts in multilingual environment, it still remains a challenging problem. Complications arise because of different writing styles and font sizes. For correctly recognizing a character, its segmentation plays an important role, as it is responsible for separating the characters from word images [1]. Sometimes multiple touched characters create problems during segmentation and consequently recognition error takes place. Moreover, Indic script like Devanagari contains vowels and consonants with modifiers, which make complicated compositions and present additional challenges for segmentation and recognition.

A brief description of Latin and Devanagari scripts is given as follows:

(i) Indian documents include English language, which is a Latin script. It contains 26 uppercase and 26 lowercase letters. Out of which, 5 letters (A, E, I, O, U) are vowels and the rest are consonants. Letters of Latin script in uppercase and lowercase are shown in Fig. 1(a).

(ii) Devanagari script has 13 vowels (Swar) and 34 consonants (Vyanjan) with 14 vowel modifiers [3].

2169-3536 © 2018 IEEE. Translations and content mining are permitted for academic research only.
Personal use is also permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

FIGURE 1. (a) Letters of Latin script in upper and lower cases (b) Vowels (Swar), Consonants (Vyanjan) and Vowel modifiers of Devanagari script.

These modifiers sometimes connect to consonants either above or below the line and make a whole letter complex. Each letter of this script has a header line called as 'Sirorekha'. Fig. 1(b) shows vowels, consonants and vowel modifiers of Devanagari script.

It is observed that most of the character segmentation and recognition methods reported in the literature are developed for specific script and text (either printed or handwritten). Furthermore, performance of Indic script OCRs is poorer than Latin script OCRs [2]. Hence, the focus of the proposed research work is to segment and recognize the characters from scanned multilingual Indian documents with intermixed texts. The main contributions of the proposed research work are as follows:

- For character segmentation, a hybrid scheme is proposed where at the beginning; pre-processing is performed on word images. Subsequently, one pixel width image, i.e. thinning operation is applied on words and projection profile is used to find primary segmentation paths. Later, distance related criteria are utilized to get fine segmentation paths. During post-processing step, over-segmented and overlapped characters are separated using graph distance theory. Finally, a trained Support Vector Machine (SVM) classifier is employed to validate the segmentation results. Proposed character segmentation algorithm is evaluated on publicly available Tobacco-800 database and proprietary database.
- In the proposed character recognition system, three new features are computed, which are based upon structural geometry of characters. For the first feature, distances between center pixel and character pixels are found out in each non-overlapping block. These non-overlapping blocks are formed with respect to center pixel of character. Later on, row wise and column wise distances are normalized. In second feature, cut vectors are generated in eight different directions. These cut vectors represent number of transitions between text and background pixels. For constructing third feature, number of eight neighbors of each character pixel is computed. Proposed character recognition algorithm is evaluated on publicly available CPAR, Dongre and Mankar, CVLSD and Chars74k databases along with the proprietary database.

The paper is structured as follows: section II presents the related works of character segmentation and recognition. Proposed character segmentation and recognition algorithms are described in section III. Experimental results are discussed in section IV and finally, conclusion is given in section V.

II. RELATED WORK

A. CHARACTER SEGMENTATION

The main aim of character segmentation is to accurately isolate the characters from a word image. These segmentation methods are broadly classified into three categories, (i) only segmentation based (ii) only recognition based and (iii) hybrid of (i) and (ii) [4].

In segmentation based category, Nomura *et al.* [1] used histogram technique for primary character segmentation. Later, morphological thickening and thinning operations are used to segment overlapped and touched characters. Garain and Chaudhuri [5] also worked for touched machine-printed characters segmentation using multi-factorial analysis. This analysis is based upon factors like transitions, blob thickness and middleness. Chen and Wang [6] and Nikolaou *et al.* [7] developed segmentation algorithms for single and multiple touched handwritten English numerals using thinning process on background and foreground regions. Tian *et al.* [8] designed a novel Harrow space filter and weighted map algorithm for license plate character segmentation. Bansal and Sinha [9] worked on Devanagari characters segmentation using projections and statistical dimensional information of the characters. Zheng *et al.* [10] used various

structural properties between background and foreground regions to detect isolated characters. Later, with vertical projection and some heuristics, touched Arabic characters are segmented. Tripathy and Pal [11] worked on text-line, word and character segmentations using water reservoir based concept for Odia text.

In recognition based category, Sharma and Dhaka [12]–[14] developed a pixel plot and trace based framework to segment the characters using artificial neural network. Grafmüller and Beyerer [15] utilized normalized projection profile and Bayes theorem for character segmentation. Here, segmentation points are determined with the help of maximum a posteriori estimation.

In hybrid based category, Rehman and Saba [4] segmented English cursive characters using their geometrical properties and ligatures. Later, neural network is used to enhance the segmentation results. Roy *et al.* [16] worked on multi-oriented touched characters segmentation. Initially, connected component analysis is performed to segment individual components, where a trained SVM is used to verify whether the segmented component is isolated or touched characters. Later, dynamic programming algorithm is employed to find out correct segmentation regions of touched characters.

Character segmentation is a very important step for OCR because its efficiency depends on proper character segmentation results [17]. It is observed that segmentation based methods are fast. However, they are efficient for particular type of script and text (e.g. either printed or handwritten) [1], [5], [6]. On the other hand, recognition based methods are highly accurate but require high resources both in terms of time and memory, as their performance depends totally upon classifiers [12]–[14]. Hybrid based systems are highly accurate and fast, as the use of classifier is limited only up to post-processing and verification [4]. These systems, by altering the classifier training, can be used for many scripts having either of the text.

B. CHARACTER RECOGNITION

Nowadays, many algorithms are available commercially for character recognition. These algorithms contain three stages, namely pre-processing, feature extraction and classification. Feature extraction is one of the core steps of recognition. In these algorithms, mainly two types of features are calculated. These features are based on texture appearance and geometrical structure of character. Texture features transform the domain of components and exhibit periodicity [18], [19]. On the contrary, structural features are dependent on character strokes, their orientations and sizes. These features provide a well-defined symbolic overview of components, which is based upon the spatial arrangement of pixels [20].

Texture features are calculated using transforms like Fourier transform [18], [21], wavelet transform [22]–[24], Gabor transform [25] and Scale Invariant Feature Transform (SIFT) [26]. Hartley transform is also utilized in [21] to compute features. Low frequency components from these transforms reflect the basic shape of the character, whereas high frequency components provide detail variations [27], [28]. Conventional transforms like Fourier and wavelet are efficient to capture details in one dimension [29]. In the context of two-dimensional signal like image, these transforms extract details when the two-dimensional signal is represented by collection of one-dimensional signals. In addition, these transforms are unable to handle smooth contours and randomly oriented edges. Gabor filter, somehow overcomes these problems but has spectral limitations. Besides these transforms, gradients features are developed in [20], [26], and [30]–[33], whereas run length features are formed in [23], [30], and [34].

Structural features are computed by Dash *et al.* [27], [28] using chords and constellation diagram of characters. Structural stroke features are calculated in [35]–[39] and [40]–[45], whereas shape features are proposed in [46]–[51]. In addition, fitting models [47] and trajectory models [45], [52]–[55] are also used to form structural features. The satisfactory performance from these features describes the importance of stroke and structural information of characters [20]. Conversely, the scope of these methods is very limited up to the characters of one or two scripts because structural features are normally script dependent.

In general, performance of recognition system depends upon the features extracted from characters images [47]–[49]. These features should uniquely classify the character in less time. Computing transform feature on the character is a tricky task due to the variation in rotation, direction and frequency [19]. Structural features are very efficient if characters of script (like Indic scripts) are rich in strokes information. If these structural features are normalized, then robustness with respect to font styles, sizes and noise is easily achieved [17], [20].

III. PROPOSED WORK

Fig. 2 presents the structure of proposed work for character segmentation and recognition. Herein, character segmentation is divided into four phases, whereas character recognition includes training and testing phases. Three features are extracted for recognizing characters.

A. PROPOSED CHARACTER SEGMENTATION ALGORITHM

The proposed character segmentation algorithm consists of pre-processing, segmentation, post-processing and post-verification using SVM to validate the segmentation results. These steps are described as follows:

1) PRE-PROCESSING

In pre-processing, gray scale word image I of size $X \times Y$ having pixel intensity $f(m, n)$ of the pixel located at (m, n) is binarized. Binarization is performed to reduce computational complexity of the algorithm, as only two colors are present for processing. Morphological erosion operation is then performed on the binarized word image to join disconnected components as shown in Fig. 3(b). These components are

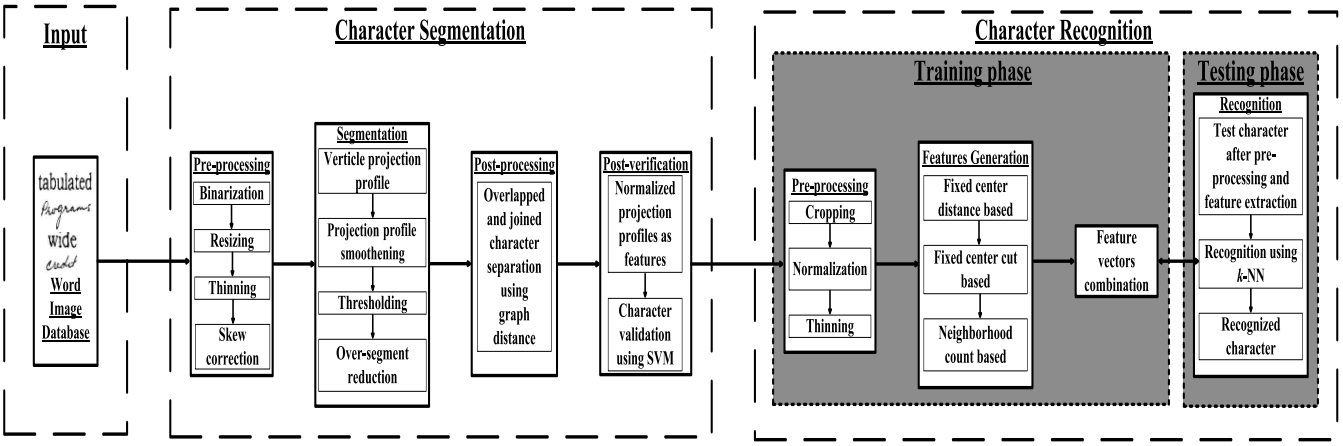

FIGURE 2. Structure of proposed work for character segmentation and recognition.

generally disconnected due to noise. Later, skew of word images is corrected and thinning operation is applied.

2) CHARACTER SEGMENTATION

Initially, vertical projection profile of thinned word image is calculated as shown in Fig. 3(d). Let, this thinned image is denoted by I_{th} of size $X \times Y$ having pixel intensity $f_{th}(m, n)$ and its vertical projection profile is given as,

$$h_n = \mathrm{Sum}_{m=1}^{X} f_{th}\,(m, n) \tag{1}$$

Here, h_n is the number of vertical projection lines varying from 1 to Y. To remove noise and some false projection lines [56], this projection profile function h_n is smoothed using Gaussian low pass filter G [57] with standard deviation σ_n in vertical direction given as,

$$G\,(\sigma_n) = \frac{1}{2\pi\sigma_n} e^{-\left(\frac{n^2}{2\sigma_n^2}\right)} \tag{2}$$

This smoothed projection profile is shown in Fig. 3(e). Now, to find primarily Segmentation Path (SP), only those projection lines are retained [4], which contain values in the range '0' to '2' (as threshold). When these projection lines are mapped on word image, over-segmentation occurs as shown in Fig. 3(f). This over-segmentation is emerged due to the connections or ligatures exist between characters and within the characters like 'r', 'm', 'n', 'u'. To reduce this over-segmentation, some of the projection lines are reduced by applying the following conditions.

(i) If difference of two projection lines is less than or equal to threshold of 3 pixels, retain the right one and remove the left one.
(ii) If difference of two projection profile lines is greater than this threshold, an average of two projection lines is calculated.
(iii) Still, if problem of over-segmentation persists and whenever difference of two projection lines is greater than threshold, retain the left one.

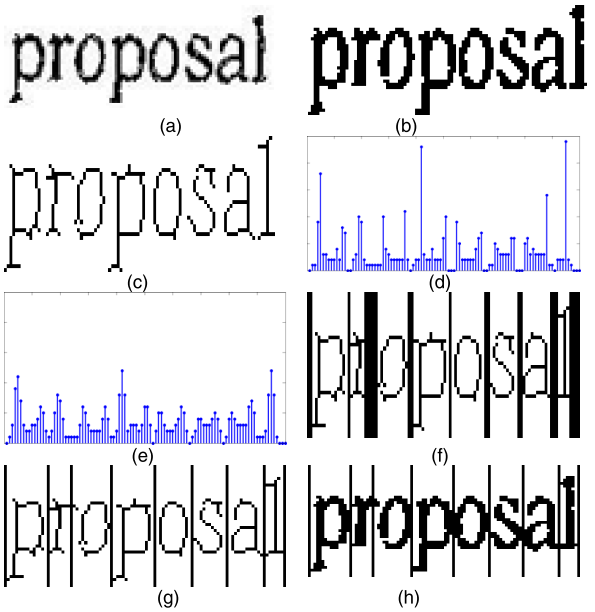

FIGURE 3. (a) Word image (b)Binarized form (c) Eroded and thinned form of binary image (d)Vertical projection profile of thinned image (e) Smoothed projection profile (f) Projections on thinned image (g) Segmentation results after applying heuristics (h) Final segmentation result.

Steps for reducing the over-segmentation are presented in Algorithm 1 and result is shown in Fig. 3(g).

3) POST-PROCESSING

Prior to this step, characters that are vertically aligned are segmented accurately. Some joined characters whose ligature contains pixels less than threshold are also segmented in precise manner. However, overlapped and joined characters whose ligatures contain pixels more than threshold are segmented using graph distance theory. For this, gray scale part of two joined characters is considered and their middle column is determined. Here, assumption is made that characters

Algorithm 1 Over-Segmentation Reduction

1. Start with smoothed projection profiles h_ns of a thinned word image,
2. **For** $n = 1$ to number of h_ns $- 1$,
3. **If** difference of two consecutive projection lines is less than or equal to threshold of 3 pixels, i.e., $h_{n+1} - h_n \leq 3$ **then**, Segmentation Path (SP) is h_{n+1} and $cnt = cnt + 1$,
4. **end if**
5. **If** difference of two consecutive projection lines is greater than threshold of 3 pixels, i.e., $h_{n+1} - h_n > 3$ and $cnt \neq 0$ **then**, Segmentation Path (SP) $= (h_{n+1} + h_n)/cnt$,
6. **end if**
7. **If** difference of two consecutive projection lines is greater than or equal to threshold of 3 pixels, i.e., $h_{n+1} - h_n \geq 3$ and $cnt = 0$ **then**, Segmentation Path (SP) is h_{n+1} and $cnt = cnt + 1$,
8. **end if**
9. **end for**

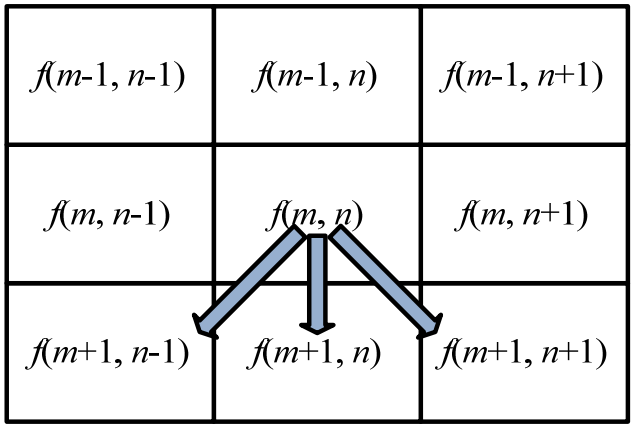

FIGURE 4. Searching mechanism for finding next segmentation pixel.

are of equal width and segmentation pixels are high intensity pixels. Now, starting in downward direction and in each iteration, present pixel becomes the center pixel of a searching window 3×3, which is shown in Fig. 4. As search is in the downward direction, only next row pixels are considered to decrease the redundancy and to make searching mechanism fast and simple. Search for the next pixel is performed using following conditions and segmentation result is shown in Fig. 5.

(i) If present pixel's intensity $f(m, n) = 255$, then pixel at location (m, n) becomes a segmentation pixel and searching mechanism will start exactly from next row and same column pixel, i.e. from the pixel at location $(m + 1, n)$.

(ii) If present pixel's intensity $f(m, n) \neq 255$, then pixel at location (m, n) becomes a segmentation pixel and searching mechanism will start from next row and any of the next column pixels, i.e. from the pixels at locations $(m + 1, n - 1)$, $(m + 1, n)$ or $(m + 1, n + 1)$. Here, these pixel positions are considered as nodes of a graph. Therefore, Euclidean distance weights from graph theory are considered for assigning priorities of next row pixels. Steps for searching a next pixel are presented in Algorithm 2.

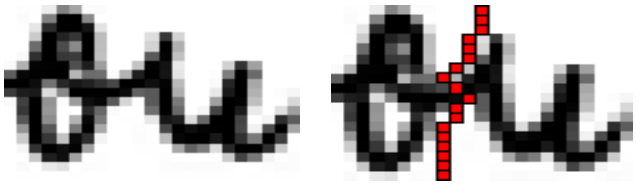

FIGURE 5. Segmentation of two joined characters using graph distance theory.

4) POST-VERIFICATION

Promising results are obtained for character segmentation ahead of this step. However, over-segmentation problem is still persists in open characters like 'n', 'v', 'm', 'w', 'u'. Therefore, to avoid this problem and validate the segmentation results, post-verification is performed using SVM classifier [58], [59]. SVM is used for binary classification problems. Suppose a training set of label pairs (x_i, y_i), $i = 1, 2, \ldots . m$ is given, where $x_i \epsilon R^n$ and $y_i \epsilon \{1, -1\}^m$. SVM maximizes the margin between classes by optimizing the hyper-plane given as,

$$\min_{w,b} \frac{1}{2} w^T w \quad \text{subject to } y_i(w^T x_i + b) \geq 1 \tag{3}$$

Here, w is weight vector and b is bias. To calculate these parameters, SVM algorithm minimizes the cost function given as,

$$\min_{w,b,\zeta} \frac{1}{2} w^T w + C \sum_{i=1}^{n} \zeta_i$$
$$\text{subject to } y_i \left(w^T x_i + b \right) \geq 1 - \zeta_i \quad \text{and } \zeta_i \geq 0 \tag{4}$$

Here, C controls tradeoff between training error and generalization, whereas ζ_i is a slack variable, which tolerate errors and need to be minimized [26]. If training vectors x_i are not separable linearly, at that time training vectors are mapped into higher dimensional space through a function $\phi(x_i)$ given in (5). In higher dimensional space, SVM finds an optimal hyper-plane with maximal margin.

$$\min_{w,b,\zeta} \frac{1}{2} w^T w + C \sum_{1=1}^{m} \zeta_i$$
$$\text{subject to } y_i \left(w^T \phi(x_i) + b \right) \geq 1 - \zeta_i \quad \text{and } \zeta_i \geq 0 \tag{5}$$
$$K\left(x_i, x_j\right) = \phi\left(x_i\right)^T \phi(x_j) \tag{6}$$
$$\text{Linear Kernel: } K\left(x_i, x_j\right) = \left(x_i\right)^T \left(x_j\right) \tag{7}$$

Equation (6) defines kernel for SVM and linear kernel is given in (7).

Algorithm 2 Searching Next Pixel

1. Start searching next pixel in downward direction in 3×3 window,
2. If intensity of present pixel $f(m, n) = 255$ **then**, assign pixel at location (m, n) as a segmentation pixel and searching will start from pixel at location $(m + 1, n)$.
3. end if
4. If intensity of present pixel $f(m, n) \neq 255$ **then**, find maximum between next row and any of the next column pixels using Euclidean distance weights, i.e., $\max\{\sqrt{2} \times f(m+1, n-1), f(m+1, n), \sqrt{2} \times f(m+1, n+1)\}$,
5. If maximum value is $\sqrt{2} \times f(m+1, n-1)$ **then**, assign pixel at location $(m+1, n-1)$ as a segmentation pixel.
6. end if
7. If maximum value is $\sqrt{2} \times f(m+1, n+1)$ **then**, assign pixel at location $(m+1, n+1)$ as a segmentation pixel.
8. end if
9. If maximum value is $f(m+1, n)$ **then**, assign pixel at location $(m+1, n)$ as a segmentation pixel.
10. end if
11. If values $\sqrt{2} \times f(m+1, n-1)$ and $f(m+1, n)$ are equal and maximum **then**, assign pixel at location $(m+1, n)$ as a segmentation pixel.
12. end if
13. If values $\sqrt{2} \times f(m+1, n+1)$ and $f(m+1, n)$ are equal and maximum **then**, assign pixel at location $(m+1, n)$ as a segmentation pixel.
14. end if
15. If values $\sqrt{2} \times f(m+1, n-1)$ and $\sqrt{2} \times f(m+1, n+1)$ are equal and maximum **then**, assign pixel at location $(m+1, n+1)$ as a segmentation pixel.
16. end if
17. If all three values, i.e. $\sqrt{2} \times f(m+1, n-1), f(m+1, n)$ and $\sqrt{2} \times f(m+1, n+1)$ are equal **then**, assign pixel at location $f(m+1, n)$ as a segmentation pixel.
18. end if
19. end if

Segmented sections f_{sect} are first categorized into two classes, i.e. correct and incorrect and are stored in training database. Features F_{1in} and F_{2im} are generated for each section to train the SVM. These features are vertical and horizontal normalized projection profiles described in Algorithm 3. For validation, features are calculated from segmented sections of a word image and are fed to the trained SVM. If valid character is not found, next segment is merged with the present one and process is repeated as shown in Fig. 6. Result after post-verification is shown in Fig. 7(b) and its steps are presented in Algorithm 3.

B. PROPOSED CHARACTER RECOGNITION ALGORITHM

Proposed character recognition algorithm contains pre-processing, features extraction and character classification steps. In pre-processing, background pixels are cropped from

Algorithm 3 Post-Verification

1. Start with segmented sections f_{sect}
2. For i = 1 to *sect*
3. Calculate vertical projection feature,
$F_{1in} = \sum_{m=1}^{X} f_{sect}(m, n)/\max(F_{1in})$,
4. Calculate horizontal projection feature,
$F_{2im} = \sum_{n=1}^{Y} f_{sect}(m, n)/\max(F_{2im})$,
5. end for
6. For i = 1 to *sect*
7. Test validity T_i of segmented sections f_{sect} using SVM, i.e.,
Test $T_i = \text{SVM}(F_{1in}, F_{2im})$,
8. If the particular segmented section is not valid, i.e. $T_i = 0$, **then**, merge next section with present T_i and test the validity of merged sections.
9. end if
10. If the particular segmented section is valid, i.e. $T_i = 1$, **then**, proceed with next segmented section to test its validity.
11. end if
12. end for

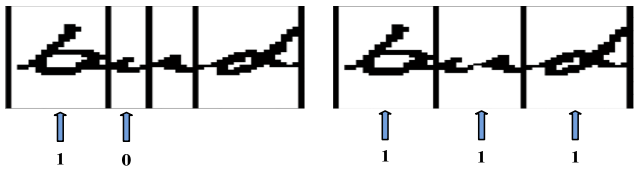

FIGURE 6. Character validation.

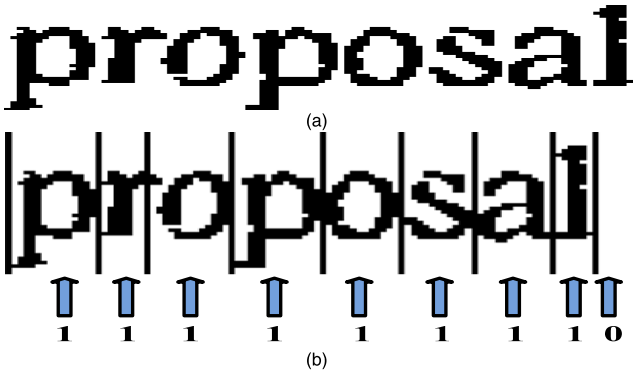

FIGURE 7. (a) Word image (b) Result after post-verification.

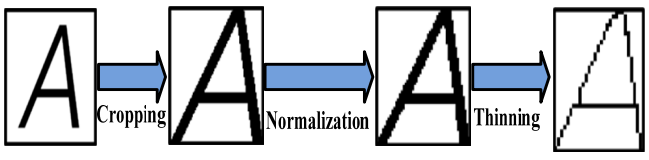

FIGURE 8. Pre-processing steps involved in character recognition.

all four sides so that first pixel from either side becomes the text pixel. Thereafter, gray level character is normalized to locate center pixel and is converted into binary form.

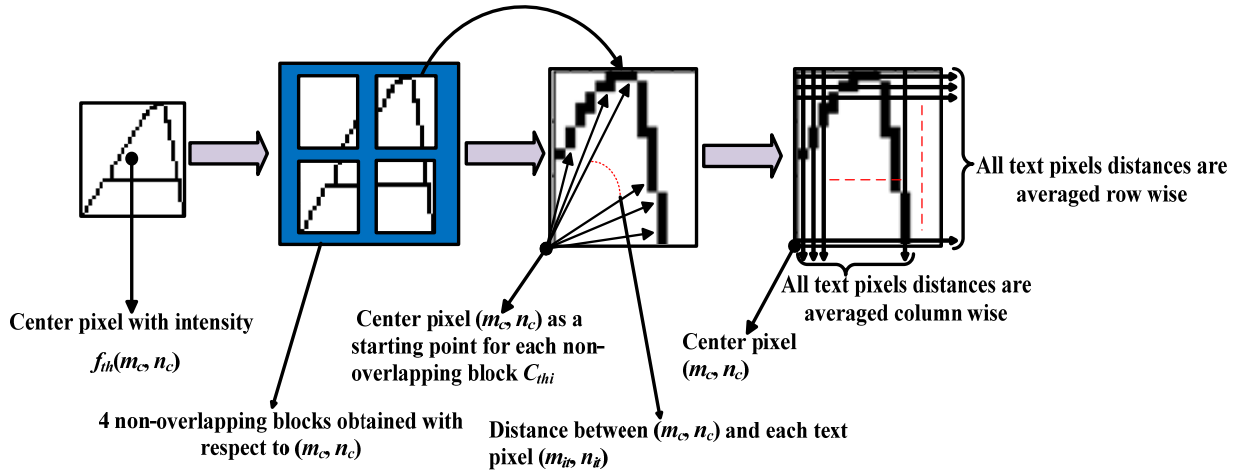

FIGURE 9. Generation of FCDF from one of the non-overlapping block.

This binary image is eroded with suitable structural mask and is thinned to make its width one text pixel. Pre-processing steps for character recognition are shown in Fig. 8. Three new structural features are computed as follows:

1) FIXED CENTER DISTANCE BASED FEATURE GENERATION (FCDF)

In this feature, thinned character image C_{th} of size $M \times N$ having pixel intensity $f_{th}(m, n)$ of the pixel (m, n) is divided into i (four in our case) non-overlapping blocks C_{thi}. These blocks are formed by keeping center pixel (m_c, n_c) of thinned image as a reference point given by (8).

$$C_{th} \Rightarrow C_{thi} \text{ with respect to } f_{th}(m_c, n_c), \begin{cases} \text{if } i = 1, m_i = 1 \text{ to } m_c \text{ and } n_i = 1 \text{ to } n_c \\ \text{if } i = 2, m_i = m_c \text{ to } M \text{ and } n_i = 1 \text{ to } n_c \\ \text{if } i = 3, m_i = 1 \text{ to } m_c \text{ and } n_i = n_c \text{ to } N \\ \text{if } i = 4, m_i = m_c \text{ to } M \text{ and } n_i = n_c \text{ to } N \end{cases} \tag{8}$$

Here, center pixel (m_c, n_c) may be either a background or a text pixel of C_{th}. Size of each block is $M_i \times N_i$ having pixel intensity $f_{thi}(m_i, n_i)$. At this moment, center pixel of thinned image is considered as a starting point of each non-overlapping block and its distance with each text pixel is calculated. This distance is Euclidean distance D_{it} defined by (9).

$$D_{it}(m_c, n_c, m_i, n_i) = \sqrt{(m_c - m_{it})^2 + (n_c - n_{it})^2} \tag{9}$$

Where, (m_{it}, n_{it}) is a text pixel in each non-overlapping block C_{thi}. Text pixels are present either single or multiple times in each row and column. Average distances of all the text pixels in a row and that of in a column from the center pixel are computed. These row wise and column wise distances are normalized and are rounded to nearest integer. Horizontal FCDF (HFCDF) and Vertical FCDF (VFCDF) defined in (10) and (11) are formed for each non-overlapping block. Fig. 9 shows generation of FCDF from one of the non-overlapping block and Algorithm 4 in Table 1 describe steps for generating FCDF. Generated FCDF feature for (4^Z) zones is FCDF $= \left\{(\text{HFCDF})_{i=1}^{4^Z}, (\text{VFCDF})_{i=1}^{4^Z}\right\}$, whereas its size is $2^Z \times (M + N)$ for $Z = 1$.

$$\text{HFCDF}_i = \text{round}\left\{\frac{Ro_{m_i}}{\max\left(Ro_{m_i}\right)}\right\}, \quad Ro_{m_i} = \left\{\mu_{n_i}\left(D_{itn_i}\right)\right\}_{m_i} \tag{10}$$

$$\text{VFCDF}_i = \text{round}\left\{\frac{Co_{n_i}}{\max\left(Co_{n_i}\right)}\right\}, \quad Co_{n_i} = \left\{\mu_{m_i}\left(D_{itm_i}\right)\right\}_{n_i} \tag{11}$$

Equations (8-11) are defined for $i = 1$ to 4^Z where $Z = 1$, C_{thi} represents four non-overlapping blocks. Ro_{m_i} and Co_{n_i} denote average distances of rows and columns from center pixel.

2) FIXED CENTER CUT BASED FEATURE GENERATION (FCCF)

In this feature, number of cuts are calculated from the thinned character image with center pixel (m_c, n_c) as a reference point in the directions described by (12). These cuts represent transition between text pixel and background pixel and vice versa as shown in Fig. 10. For eight different scan lines, generated FCCF is

$$\text{FCCF} = \left\{\begin{matrix}\text{UFCCF,LOFCCF,RFCCF,LEFCCF,} \\ \text{URDFCCF,ULDFCCF,LRDFCCF,LLDFCCF}\end{matrix}\right\}$$

and its size is eight. Here, UFCCF, LOFCCF, RFCCF, LEFCCF, URDFCCF, ULDFCCF, LRDFCCF, LLDFCCF denote upper FCCF, lower FCCF, right FCCF, left FCCF, upper right diagonal FCCF, upper left diagonal FCCF, lower

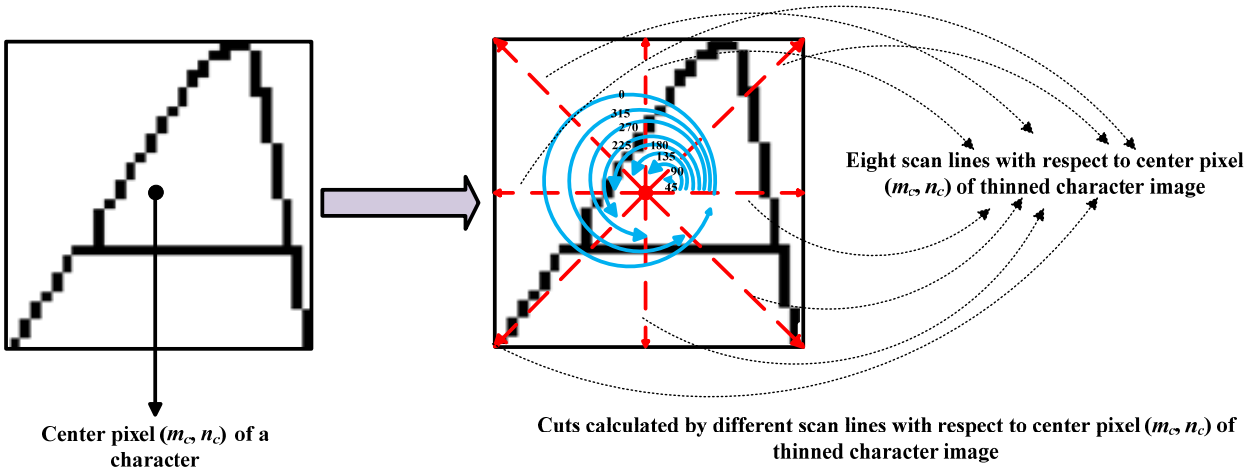

FIGURE 10. Generation of FCCF from eight scan lines.

right diagonal FCCF and lower left diagonal FCCF, respectively and represent number of cuts generated in the directions described by (12). Fig. 10 shows generation of FCCF and Algorithm 5 in Table 1 describe steps for generating FCCF.

$$\left.\begin{aligned}
&\text{UFCCF} = \sum_{i=1}^{N_{C90^0}} CU_{90^0}, \text{LOFCCF} = \sum_{i=1}^{N_{C270^0}} CU_{270^0},\\
&\text{RFCCF} = \sum_{i=1}^{N_{C0^0}} CU_{0^0}, \text{LEFCCF} = \sum_{i=1}^{N_{C180^0}} CU_{180^0},\\
&\text{URDFCCF} = \sum_{i=1}^{N_{C135^0}} CU_{135^0}, \text{ULDFCCF} = \sum_{i=1}^{N_{C45^0}} CU_{45^0},\\
&\text{LRDFCCF} = \sum_{i=1}^{N_{C315^0}} CU_{315^0}, \text{LLDFCCF} = \sum_{i=1}^{N_{C225^0}} CU_{225^0}
\end{aligned}\right\} \quad (12)$$

Here, N_C is the number of pixels at scan angles defined in (12) with respect to (m_c, n_c) and CU is the number of cuts or transitions.

3) NEIGHBORHOOD COUNTS BASED FEATURE GENERATION (NCF)

In this feature, thinned image C_{th} is considered again and number of neighborhood character pixels for each text pixel $f_{tht}(m, n)$ are calculated, which is described by (13). Each text pixel is centered in a window of size 3×3. Generated NCF and its size are

$$\text{NCF} = \left\{\begin{aligned}
&A(m-1, n-1), A(m-1, n),\\
&A(m-1, n+1), A(m, n-1),\\
&A(m, n+1), A(m+1, n-1),\\
&A(m+1, n), A(m+1, n+1)
\end{aligned}\right\}$$

and eight, respectively. Fig. 11 shows the generation of NCF where Algorithm 6 in Table 1 describe steps for

TABLE 1. Steps for generating FCDF, FCCF and NCF.

Algorithm 4 FCDF generation
1. Create 4^Z non-overlapping blocks, with (m_c, n_c) as a reference point, where Z=1.
2. Compute Euclidean distance between (m_c, n_c) and (m_{it}, n_{it}) in each C_{thi}.
3. Calculate average distances in horizontal and vertical directions.
4. Normalized and rounded the distances to nearest integer.
5. Generate FCDF = $\{(\text{HFCDF})_{i=1}^{4^Z}, (\text{VFCDF})_{i=1}^{4^Z}\}$, where Z=1.
Algorithm 5 FCCF generation
1. Create eight scan lines with (m_c, n_c) as a reference point.
2. Calculate cuts for different scan lines in the direction given in (12).
3. Generate FCCF = $\left\{\begin{matrix}\text{UFCCF, LOFCCF, RFCCF, LEFCCF,}\\ \text{URDFCCF, ULDFCCF, LRDFCCF, LLDFCCF}\end{matrix}\right\}$.
Algorithm 6 NCF generation
1. Store all the character pixels in the array.
2. Start with each $f_{tht}(m, n)$, calculate all its neighborhood character pixels using (13).
3. Generate NCF = $\left\{\begin{matrix}A(m-1, n-1), A(m-1, n),\\ A(m-1, n+1), A(m, n-1),\\ A(m, n+1), A(m+1, n-1),\\ A(m+1, n), A(m+1, n+1)\end{matrix}\right\}$

generating NCF.

$$\left.\begin{aligned}
&A(m-1, n-1) = \sum f_{tht}(m-1, n-1),\\
&f_{tht}(m-1, n-1) \text{ and } f_{tht}(m, n) \text{ are text pixels}\\
&A(m-1, n) = \sum f_{tht}(m-1, n),\\
&f_{tht}(m-1, n) \text{ and } f_{tht}(m, n) \text{ are text pixels}\\
&A(m-1, n+1) = \sum f_{tht}(m-1, n+1),\\
&f_{tht}(m-1, n+1) \text{ and } f_{tht}(m, n) \text{ are text pixels}\\
&A(m, n-1) = \sum f_{tht}(m, n-1),\\
&f_{tht}(m, n-1) \text{ and } f_{tht}(m, n) \text{ are text pixels}\\
&A(m, n+1) = \sum f_{tht}(m, n+1),\\
&f_{tht}(m, n+1) \text{ and } f_{tht}(m, n) \text{ are text pixel}\\
&A(m+1, n-1) = \sum f_{tht}(m+1, n-1),\\
&f_{tht}(m+1, n-1) \text{ and } f_{tht}(m, n) \text{ are text pixels}\\
&A(m+1, n) = \sum f_{tht}(m+1, n),\\
&f_{tht}(m+1, n) \text{ and } f_{tht}(m, n) \text{ are text pixels}\\
&A(m+1, n+1) = \sum f_{tht}(m+1, n+1),\\
&f_{tht}(m+1, n+1) \text{ and } f_{tht}(m, n) \text{ are text pixels}
\end{aligned}\right\} \quad (13)$$

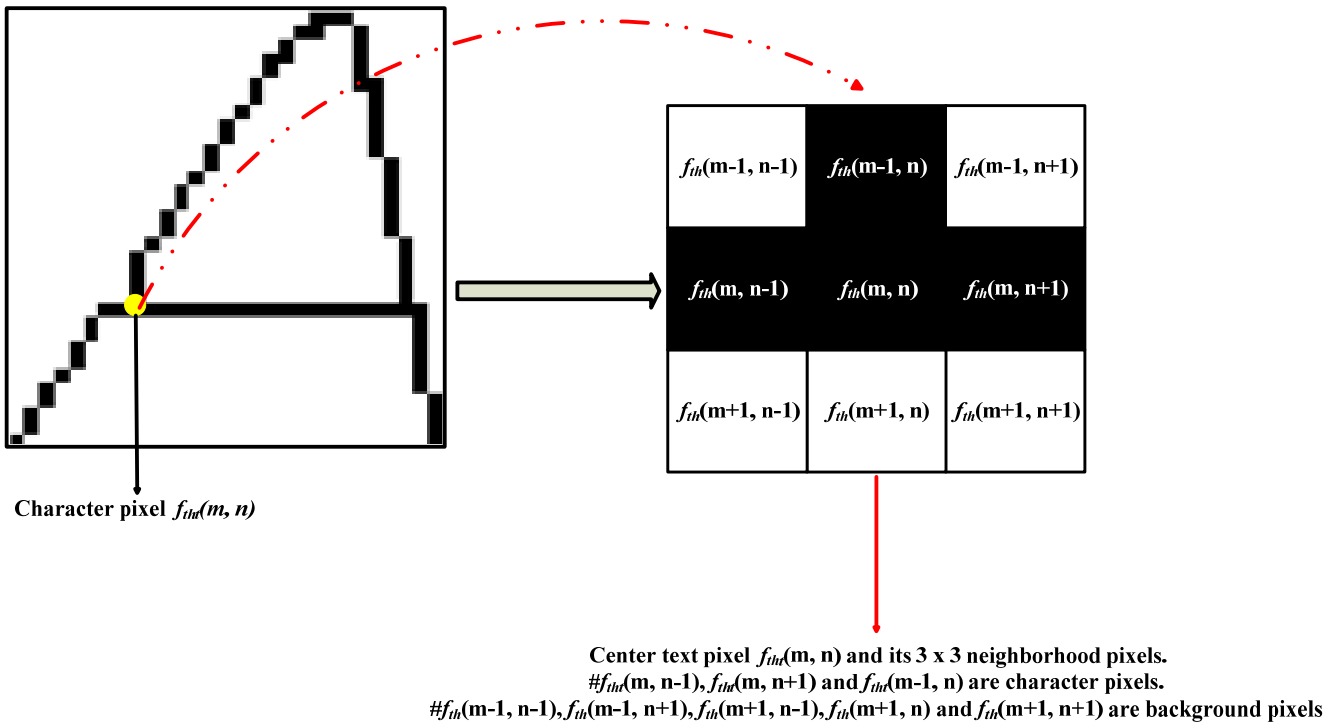

FIGURE 11. Generation of NCF.

Here, $A\,(m-1:m+1, n-1:n+1)$ are the summations of $f_{tht}\,(m-1:m+1, n-1:n+1)$ and the conditions are specified in (13).

4) CLASSIFICATION

Subsequent to feature generation, next step is to provide class membership to each character that comes for classification. For this decision making step, k-Nearest Neighbor (k-NN) is used in the proposed character recognition work where features of input character is compared with features of training samples to compute similar k neighbors. Accuracy of k-NN depends upon two factors, (i) distance function used and (ii) value of k. Here, three distances, i.e. Euclidean, city block and cosine are used to calculate distance between an input feature vector a and each training feature vector b. Euclidean distance is given as,

$$D\,(a, b) = \sqrt{\sum_{i=1}^{N} (a_i - b_i)^2} \tag{14}$$

City block distance is given as,

$$C\,(a, b) = \sum_{i=1}^{N} |a_i - b_i| \tag{15}$$

Cosine distance is given as,

$$C_{cos}\,(a, b) = \frac{\sum_{i=1}^{N} a_i b_i}{\sqrt{\sum_{i=1}^{N} a_i^2}\sqrt{\sum_{i=1}^{N} b_i^2}} \tag{16}$$

Here, N denotes dimension of feature vectors and distances (D, C, and C_{cos}) between input vector a and b_i are calculated using (14-16). At this instant, k nearest distances are taken into account and number of times input vector belonging to each class are calculated. Output of classifier is the class that occurs very frequently.

IV. EXPERIMENTAL RESULTS

To validate the proposed character segmentation and recognition algorithms, experiments are performed on publicly available database and proprietary database. For proposed character segmentation algorithm, 10-fold cross validation with multiple runs methodology is adopted. Later, final accuracy is computed by averaging the accuracies from each fold.

A. DATABASES USED

For evaluating proposed character segmentation algorithm, Tobacco-800 [60]–[62] and proprietary databases are used. Tobacco-800 database contains 1290 document images in TIFF format. Image size is varying from 1200 × 1600 to 2500 × 3200 pixels and scanned at 150-300 dpi. This database consists of documents related to master settlement agreements from Tobacco companies and research industries.

Proprietary database consists of 350 documents collected from government offices, which are generally notices and receipts. These documents contain intermixed texts in Devanagari and Latin scripts scanned at 300 dpi in TIFF format. Noise is introduced in these documents during scanning because of ink spreading and vibrations of the scanner. The different characters of variable font styles and sizes formed

printed text. Contrary, handwritten text is formed by diverse human handwritings with different writing apparatus. This handwritten text is characterized by uneven thickness and spacing between characters.

For character recognition of Latin script, Chars74k [63], CVLSD [64] and proprietary databases are used. Chars74k database contains numerals, uppercase and lowercase characters that make total 62 classes and images are in PNG format. CVLSD database consists of numerals (10 classes) only, for which 303 writers contributed. Training set contains 7000 digits and evaluation set consists of 21780 digits.

For Devanagari characters recognition, CPAR [65], Dongre and Mankar [66] and proprietary databases are used. CPAR database contains 35,000 isolated handwritten numerals and 83,300 characters. This database is collected from individuals of different ages, religions and educational backgrounds. Dongre and Mankar database contains TIFF format images of 5137 numerals and 20305 characters collected by 750 writers. All experiments are performed in MATLAB environment with Windows 7 operating system.

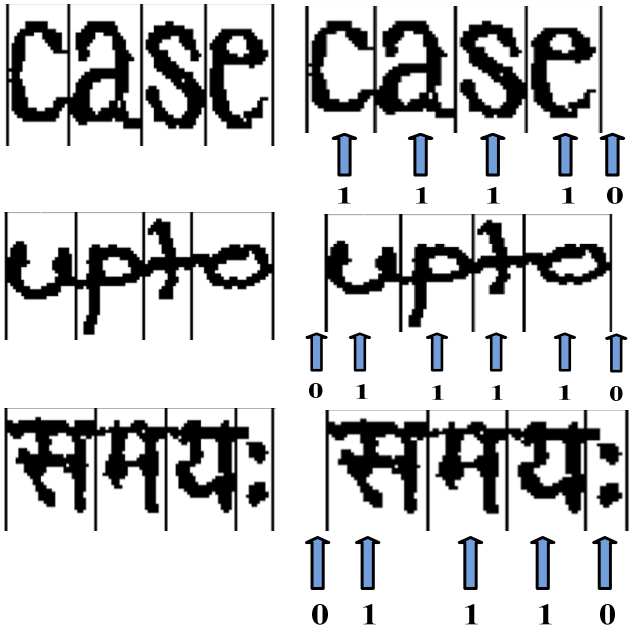

FIGURE 12. Word images and corresponding character segmentation result.

B. PERFORMANCE EVALUATION OF CHARACTER SEGMENTATION ALGORITHM

During pre-processing, size of structure element for erosion operation is kept 3×3 and threshold for Devanagari script is set to 5. While post-processing, Euclidean distance metric is used because it represents real distance between two nodes. In post-verification step, SVM with linear kernel is employed, as it distinguishes correct and incorrect sections easily and linear kernel is less complex. Fig. 12 shows some results of proposed approach. Criteria to evaluate performance of proposed character segmentation algorithm are Segmentation Rate (SR), Over-Segmentation Rate (OSR) and Bad Segmentation Rate (BSR). These are defined as follows:

$$\text{SR} = \frac{\text{Correctly segmented characters}}{\text{Total number of characters}} \times 100 \quad (17)$$

$$\text{OSR} = \frac{\text{Over-segmented characters}}{\text{Total number of characters}} \times 100 \quad (18)$$

$$\text{BSR} = \frac{\text{Errorly segmented characters}}{\text{Total number of characters}} \times 100 \quad (19)$$

Herein the proposed algorithm, if a character is segmented more than its actual boundaries, it is known as over-segmentation. If SP does not separate two characters perfectly, it is termed as bad segmentation. Instead, SP includes some part of adjacent character while segmentation and creates incorrect segmented characters. Segmentation results are reported in Table 2. For Tobacco-800 database, obtained SR is 97.71%, whereas for proprietary database, 98.86% and 97.26% SRs are obtained on Latin and Devanagari scripts, respectively. It is observed that BSRs are obtained in the range 0.96-2.54%, which are marginally higher than OSRs. The reasons are sometime segmentation algorithm could not able to locate accurate boundaries between characters during post-processing and vertical cuts are obtained for overlapped characters.

TABLE 2. Segmentation results of proposed character segmentation algorithm.

Database	SR%	OSR%	BSR%
Tobacco-800	97.71	0.22	2.07
Proprietary (Latin script)	98.86	0.18	0.96
Proprietary (Devanagari script)	97.26	0.20	2.54

The advantage of proposed character segmentation scheme is that two characters are still segmented having tight joints with no ligature. If Gaussian low pass filtering is performed during pre-processing, problem of noise in characters will get resolved and it does not require additional step. Due to touched characters, bad and missed segmentations occur that are handled in post-processing step. Further, over-segmentation problem is resolved by character validation using SVM. Some failure cases of proposed approach are shown in Fig. 13.

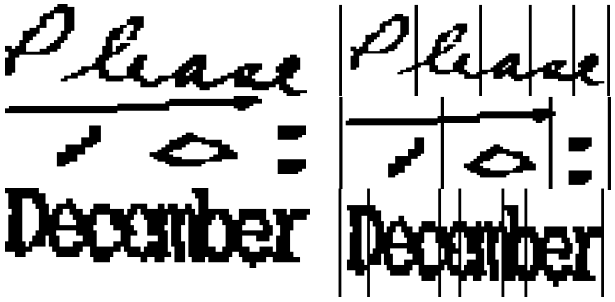

FIGURE 13. Word images and incorrect character segmentation results.

Table 3 clearly shows the superiority of the proposed character segmentation algorithm over different approaches.

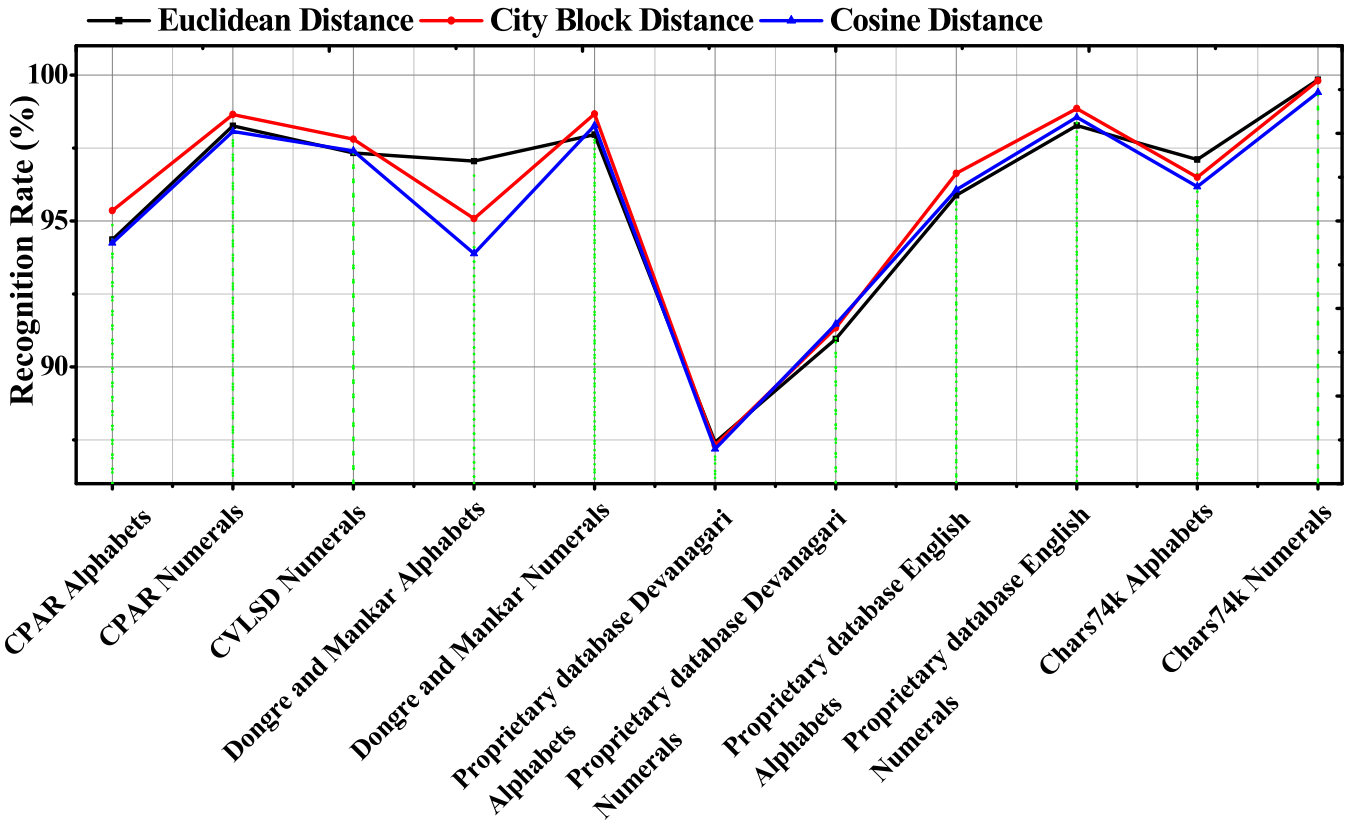

FIGURE 14. Fold-10 cross validation comparison of classification rates using k-NN with Euclidean, city block and cosine distances.

TABLE 3. Performance comparison of proposed character segmentation algorithm with other approaches

Approaches	Database	Number of Documents	SR%	Average Processing Time in Second
Rehman and Saba [4]	Tobacco-800	1290	89.91	1.45
Sharma and Dhaka [12]	Tobacco-800	1290	95.32	1.51
Dhaka Sharma [13]	Tobacco-800	1290	95.87	1.46
Sharma and Dhaka [14]	Tobacco-800	1290	96.67	1.54
Proposed	**Tobacco-800**	**1290**	**97.71**	**1.42**
Rehman and Saba [4]	Proprietary	350	90.28	1.48
Sharma and Dhaka [12]	Proprietary	350	96.42	1.49
Dhaka Sharma [13]	Proprietary	350	96.88	1.53
Sharma and Dhaka [14]	Proprietary	350	97.73	1.58
Proposed	**Proprietary**	**350**	**98.06**	**1.48**

For evaluation on Tobacco-800 and proprietary databases, only those approaches are selected that have the integration of some intelligent schemes like neural networks, SVM and so on with segmentation to enhance the accuracy. All these approaches are evaluated on 1290 documents of Tobacco-800 database and 350 documents of proprietary database. Rehman and Saba [4] obtained 89.91% and 90.28% SRs on Tobacco-800 and proprietary databases, respectively. In the same way, Dhaka and Sharma [13] attained 95.87% and 96.88% SRs on corresponding Tobacco-800 and proprietary databases. Finally, Sharma and Dhaka [12], [14] obtained average SRs of 96.00% and 97.08% on Tobacco-800 and proprietary databases, respectively. An average processing time consumed by proposed character segmentation algorithm is 1.5 second per word on computer system having 4 GB RAM, INTEL i5-3470 CPU @ 3.20 GHz.

C. PERFORMANCE EVALUATION OF CHARACTER RECOGNITION ALGORITHM

During pre-processing step, size of each character is reformed using bi-cubic interpolation method and size of structure element for erosion operation is kept 3×3. For forming FCDF, physical distance between center pixel and text pixel is required and therefore, Euclidean distance is used. For 4^Z number of zones, $Z = 1$ is considered which gives total 4 number of zones within the image. Consequently, in the formation of FCDF, 4 vectors are contributed from HFCDF and 4 vectors are contributed from VFCDF. Fixed 8 values are contributed each from FCCF (1 value from each scan line) and NCF (1 value from each neighbor). k-NN is selected for recognizing the character because k-NN has zero

TABLE 4. Fold-10 cross validation performance of proposed recognition algorithm using *k*-NN with Euclidean distance.

Database	Database Type	FCDF+FCCF+NCF RRs%									
		Set 1	Set 2	Set 3	Set 4	Set 5	Set 6	Set 7	Set 8	Set 9	Set 10
CPAR	Handwritten (Alphabets)	94.58	94.49	94.40	94.24	94.07	94.30	94.53	94.38	94.23	94.41
CPAR	Handwritten (Numerals)	98.34	98.30	98.26	98.07	97.87	98.12	98.36	98.41	98.45	98.40
CVLSD	Handwritten (Numerals)	97.47	96.97	96.47	97.30	98.13	98.03	97.93	97.30	96.67	97.07
Dongre and Mankar	Handwritten (Alphabets)	96.68	96.21	95.73	96.51	97.28	97.47	97.65	97.79	97.92	97.30
Dongre and Mankar	Handwritten (Numerals)	98.14	97.94	97.73	97.83	97.93	97.87	97.80	98.00	98.20	98.17
Proprietary Devanagari	Printed and Handwritten (Alphabets)	87.41	87.42	87.43	87.44	87.44	87.46	87.48	87.40	87.31	87.36
Proprietary Devanagari	Printed and Handwritten (Numerals)	91.4	91.40	91.40	91.10	90.80	90.77	90.73	90.60	90.47	90.94
Proprietary Latin	Printed and Handwritten (Alphabets)	96.07	95.96	95.84	95.83	95.81	95.86	95.90	95.84	95.77	95.92
Proprietary Latin	Printed and Handwritten (Numerals)	98.39	98.37	98.35	98.40	98.45	98.50	98.55	98.08	97.60	98.00
Chars74k	Printed and Handwritten (Alphabets)	97.01	97.31	97.60	97.53	97.46	96.97	96.48	96.70	96.92	96.97
Chars74k	Printed and Handwritten (Numerals)	99.80	99.85	99.90	99.95	100	99.90	99.80	99.75	99.70	99.75

training time intrinsically. In addition, it has only requirement to store the training set a priori for classification task. In the evaluation of proposed character recognition algorithm, Recognition Rate (RR) is used, which is given as,

$$\text{RR} = \frac{\text{Number of correctly classified samples}}{\text{Total number of samples}} \times 100 \quad (20)$$

Proposed features can distinguish between two similar appearing characters to an extent. As, characters contain structurally different distances between text and center pixels along with different number of neighbors. In addition, proposed FCCF is robust to character scaling because number of cuts in each half remain same even if character dimensions are varied.

Table 4 shows 10-fold cross validation recognition accuracies of the proposed character recognition scheme over different databases with four runs. Here, *k*-NN is used with Euclidean distance. Four runs are selected because it is noticed that increasing runs after four does not provide much improvement in accuracy. For 10-fold cross validation recognition accuracy, database is divided into ten equal parts. One of its parts is considered for testing and remaining nine subsets are considered for training. The distribution of training and testing images from different databases is kept in the ratio of 90:10. For CPAR database, out of 83,300 character images, 74,970 images are used for training and 8,330 images are used for testing. It is observed that, 2.7-12.4% higher RRs are obtained on numeral databases, which is due to the presence of less number of overlapping classes.

Table 5 illustrates the average performance of proposed features in individual and combined forms using *k*-NN with Euclidean distance. From this table, it is observed that FCCF is showing slightly less accuracy (on an average 7-10%) than FCDF and NCF, however, high accuracy is obtained in combined form. Overall accuracy of the proposed scheme is marginally less in the range 87.42-90.96% for proprietary database of Devanagari script; otherwise, prominent RRs are obtained. The reason is due to the presence of smeared and compound Devanagari characters. It is seen that some misclassifications occur between 'I' and '1' and '0', 'o' and 'O'. These misclassifications are removed by using local structural features [46], [51]. To differentiate between 'I' and '1', number of text pixels at top left corner of character are calculated, whereas aspect ratio is calculated to differentiate the characters '0', 'o' and 'O'.

Fig. 14 presents 10-fold cross validation RRs comparison over different databases using Euclidean, city block and cosine distances. It is seen that comparatively higher rates (in the range 0.2-0.5%) are obtained with city block distance as compared to other distances.

Table 6 shows comparative analysis of the proposed algorithm with other approaches. Table also consists of additional information as feature types, decision schemes, output classes and databases used by various methods. Proposed algorithm gives better RRs than other approaches on both numerals and alphabets databases. Highest RR of 98.26% is obtained from the proposed algorithm on CPAR database containing 59 classes of Devanagari script. Average recognition time

TABLE 5. Performance of proposed character recognition algorithm using *k*-NN with Euclidean distance.

Database	FCDF%	FCCF%	NCF%	Combined%
CPAR (Alphabets)	93.64	84.56	91.58	94.36
CPAR (Numerals)	98.16	81.82	93.13	98.26
CVLSD (Numerals)	97.13	80.00	90.07	97.33
Dongre and Mankar (Alphabets)	93.92	83.72	91.29	97.05
Dongre and Mankar (Numerals)	97.87	79.47	89.87	97.96
Proprietary Devanagari (Alphabets)	87.20	80.98	87.06	87.42
Proprietary Devanagari (Numerals)	90.93	89.20	89.13	90.96
Proprietary Latin (Alphabets)	95.72	87.38	93.06	95.88
Proprietary Latin (Numerals)	98.00	96.80	94.25	98.27
Chars74k (Alphabets)	95.96	83.06	92.52	97.10
Chars74k (Numerals)	99.60	84.40	96.30	99.84

TABLE 6. Comparative analysis of proposed recognition algorithm on different databases.

Algorithm	Database	Features Types	Decision Scheme	Number of Output Classes	Recognition Time in Second	RR%
Shi et al. [20]	CVLSD	Stroke detectors and structural	SVM	10	2.52	96.69
Proposed	**CVLSD**	**FCDF+FCCF+NCF**	***k*-NN**	**10**	**2.71**	**97.33**
Kumar et al. [65]	CPAR	Intensity based	*k*-NN	59	3.240	97.87
Kumar and Ravulakollu [24]	CPAR	Transform based	*k*-NN	59	3.296	98.02
Proposed	**CPAR**	**FCDF+FCCF+NCF**	***k*-NN**	**59**	**3.252**	**98.26**
Dongre and Mankar [67]	Dongre and Mankar	Geometrical	Statistical	59	4.835	72.87
Khanduja et al. [45]	Dongre and Mankar	Structural and curve fitting	Neural network	59	5.483	91.4
Proposed	**Dongre and Mankar**	**FCDF+FCCF+NCF**	***k*-NN**	**59**	**4.891**	**97.96**
Shi et al. [20]	Chars74k	Stroke detectors and structural	SVM	62	6.458	71.8
Tian et al. [33]	Chars74k	Co-HOG and ConvCoHOG	CNN	62	6.247	76.6
Proposed	**Chars74k**	**FCDF+FCCF+NCF**	***k*-NN**	**62**	**5.881**	**97.10**

per test image taken by proposed algorithm on this database is 3.252 second. Kumar and Ravulakollu [24] used various transforms for extracting features and obtained 2nd highest RR of 98.02% with average recognition time 3.296 second. Another Devanagari script database, which is developed by Dongre and Mankar where proposed algorithm has RR of 97.96%. On this database, 25.61% more RR is obtained by proposed algorithm than Dongre and Mankar [67].

On Chars74k Latin script database, RR of 97.1% is obtained by proposed algorithm on 62 classes with average recognition time 5.881 second. This RR is 26.06% and 21.11% higher than approaches proposed by Shi *et al.* [20] and Tian *et al.* [33], respectively. CVLSD database contains only 10 numerals classes where RR of 97.33% is obtained by proposed algorithm. At a glance, proposed algorithm outperforms other approaches in terms of recognition, whereas recognition times are comparable.

V. CONCLUSION

In this paper, two new algorithms are proposed for character segmentation and recognition for multilingual Indic documents consisting of printed and handwritten texts. Character segmentation is one of the important steps before OCR. Herein, heuristic based algorithm integrated with SVM is proposed for segmenting characters. Overlapped and over-segmented characters are also separated accurately during post-processing and post-verification stages, respectively. Highest SR of 98.86% is obtained on proprietary database of Latin script. For character recognition, three new structural geometry based features are proposed. FCDF and FCCF are calculated with respect to center pixel of thinned character image, whereas NCF is calculated using neighborhood information of text pixels. Proposed recognition algorithm shows highest accuracy of 99.84% on Chars74k numerals database. Comparatively 0.2-0.5% higher RRs are obtained when *k*-NN

is used with city block distance relative to other distances. Proposed algorithm is 2.7-12.4% more efficient on numerals databases as compared to databases contain alphabets.

REFERENCES

[1] S. Nomura, K. Yamanaka, O. Katai, H. Kawakami, and T. Shiose, "A novel adaptive morphological approach for degraded character image segmentation," *Pattern Recognit.*, vol. 38, no. 11, pp. 1961–1975, Nov. 2005.

[2] M. Mathew, A. K. Singh, and C. V. Jawahar, "Multilingual OCR for indic scripts," in *Proc. 12th IAPR Workshop Document Anal. Syst.*, Santorini, Greece, Apr. 2016, pp. 186–191.

[3] R. Jayadevan, S. R. Kolhe, P. M. Patil, and U. Pal, "Offline recognition of Devanagari script: A survey," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 41, no. 6, pp. 782–796, Jan. 2011.

[4] A. Rehman and T. Saba, "Performance analysis of character segmentation approach for cursive script recognition on benchmark database," *Digit. Signal Process.*, vol. 21, no. 3, pp. 486–490, May 2011.

[5] U. Garain and B. B. Chaudhuri, "Segmentation of touching characters in printed Devnagari and Bangla scripts using fuzzy multifactorial analysis," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 32, no. 4, pp. 449–459, Nov. 2002.

[6] Y.-K. Chen and J.-F. Wang, "Segmentation of single- or multiple-touching handwritten numeral string using background and foreground analysis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 11, pp. 1304–1317, Nov. 2000.

[7] N. Nikolaou, M. Makridis, B. Gatos, N. Stamatopoulos, and N. Papamarkos, "Segmentation of historical machine-printed documents using adaptive run length smoothing and skeleton segmentation paths," *Image Vis. Comput.*, vol. 28, no. 4, pp. 590–604, Apr. 2010.

[8] J. Tian, R. Wang, G. Wang, J. Liu, and Y. Xia, "A two-stage character segmentation method for Chinese license plate," *Comput. Elect. Eng.*, vol. 46, pp. 539–553, Aug. 2015.

[9] V. Bansal and R. M. K. Sinha, "Segmentation of touching and fused Devanagari characters," *Pattern Recognit.*, vol. 35, no. 4, pp. 875–893, Apr. 2002.

[10] L. Zheng, A. H. Hassin, and X. Tang, "A new algorithm for machine printed Arabic character segmentation," *Pattern. Recogn. Lett.*, vol. 25, no. 15, pp. 1723–1729, Nov. 2004.

[11] N. Tripathy and U. Pal, "Handwriting segmentation of unconstrained Oriya text," *Sadhana*, vol. 31, no. 6, pp. 755–769, Dec. 2006.

[12] M. K. Sharma and V. P. Dhaka, "Pixel plot and trace based segmentation method for bilingual handwritten scripts using feedforward neural network," *Neural Comput. Appl.*, vol. 27, no. 7, pp. 1817–1829, Oct. 2016.

[13] V. P. Dhaka and M. K. Sharma, "An efficient segmentation technique for Devanagari offline handwritten scripts using the feedforward neural network," *Neural Comput. Appl.*, vol. 26, no. 8, pp. 1881–1893, Nov. 2015.

[14] M. K. Sharma and V. P. Dhaka, "Segmentation of English offline handwritten cursive scripts using a feedforward neural network," *Neural Comput. Appl.*, vol. 27, no. 5, pp. 1369–1379, Jul. 2016.

[15] M. Grafmüller and J. Beyerer, "Performance improvement of character recognition in industrial applications using prior knowledge for more reliable segmentation," *Expert Syst. Appl.*, vol. 40, no. 17, pp. 6955–6963, Dec. 2013.

[16] P. P. Roy, U. Pal, J. Lladós, and M. Delalandre, "Multi-oriented touching text character segmentation in graphical documents using dynamic programming," *Pattern Recognit.*, vol. 45, no. 5, pp. 1972–1983, May 2012.

[17] P. Sahare and S. B. Dhok, "Review of text extraction algorithms for scene-text and document images," *IETE Tech. Rev.*, vol. 34, no. 2, pp. 144–164, Apr. 2016.

[18] V. N. M. Aradhya, G. H. Kumar, and S. Noushath, "Multilingual OCR system for South Indian scripts and English documents: An approach based on Fourier transform and principal component analysis," *Eng. Appl. Artif. Intell.*, vol. 21, no. 4, pp. 658–668, Jun. 2008.

[19] P. Sahare and S. B. Dhok, "Script identification algorithms: A survey," *Int. J. Multimedia Inf. Retr.*, vol. 6, no. 3, pp. 211–232, Sep. 2017.

[20] C.-Z. Shi, S. Gao, M.-T. Liu, C.-Z. Qi, C.-H. Wang, and B.-H. Xiao, "Stroke detector and structure based models for character recognition: A comparative study," *IEEE Trans. Image Process.*, vol. 24, no. 12, pp. 4952–4964, Dec. 2015.

[21] S. A. Mahmoud and A. S. Mahmoud, "The use of Hartley transform in OCR with application to printed Arabic character recognition," *Pattern Anal. Appl.*, vol. 12, pp. 353–365, Dec. 2009.

[22] B. P. Chacko, V. R. Vimal Krishnan, G. Raju, and P. Babu Anto, "Handwritten character recognition using wavelet energy and extreme learning machine," *Int. J. Mach. Learn. Cybern.*, vol. 3, no. 2, pp. 149–161, Jun. 2012.

[23] U. Bhattacharya and B. B. Chaudhuri, "Handwritten numeral databases of indian scripts and multistage recognition of mixed numerals," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 3, pp. 444–457, Mar. 2009.

[24] R. Kumar and K. K. Ravulakollu, "Offline handwritten DEVNAGARI digit recognition," *ARPN J. Eng. Appl. Sci.*, vol. 9, no. 2, pp. 109–115, Feb. 2014.

[25] J. R. Prasad and U. Kulkarni, "Gujrati character recognition using weighted k-NN and Mean X^2 distance measure," *Int. J. Mach. Learn. Cybern.*, vol. 6, no. 1, pp. 69–82, Feb. 2015.

[26] O. Surinta, M. F. Karaaba, L. R. B. Schomaker, and M. A. Wiering, "Recognition of handwritten characters using local gradient feature descriptors," *Eng. Appl. Artif. Intell.*, vol. 45, pp. 405–414, Oct. 2015.

[27] K. S. Dash, N. B. Puhan, and G. Panda, "Unconstrained handwritten digit recognition using perceptual shape primitives," in *Pattern Analysis and Applications*. London, U.K.: Springer, Nov. 2016, pp. 1–24. [Online]. Available: https://doi.org/10.1007/s10044-016-0586-3

[28] K. S. Dash, N. B. Puhan, and G. Panda, "BESAC: Binary external symmetry axis constellation for unconstrained handwritten character recognition," *Pattern Recognit. Lett.*, vol. 83, no. 3, pp. 413–422, Nov. 2016.

[29] K. S. Dash, N. B. Puhan, and G. Panda, "Handwritten numeral recognition using non-redundant Stockwell transform and bio-inspired optimal zoning," *IET Image Process.*, vol. 9, no. 10, pp. 874–882, Sep. 2015.

[30] G. Raju, B. S. Moni, and M. S. Nair, "A novel handwritten character recognition system using gradient based features and run length count," *Sadhana*, vol. 39, no. 6, pp. 1333–1355, Dec. 2014.

[31] P. Singh, A. Verma, and N. S. Chaudhari, "Feature selection based classifier combination approach for handwritten Devanagari numeral recognition," *Sadhana*, vol. 40, no. 6, pp. 1701–1714, Sep. 2015.

[32] A. Rahman and B. Verma, "Effect of ensemble classifier composition on offline cursive character recognition," *Inf. Process., Manage.*, vol. 49, no. 4, pp. 852–864, Jul. 2013.

[33] S. Tian *et al.*, "Multilingual scene character recognition with co-occurrence of histogram of oriented gradients," *Pattern Recognit.*, vol. 51, pp. 125–134, Mar. 2016.

[34] S. Raja and M. John, "A novel tamil character recognition using decision tree classifier," *IETE J. Res.*, vol. 59, no. 5, pp. 569–575, Sep. 2014.

[35] Y. Wen, Y. Lu, J. Yan, Z. Zhou, K. M. von Deneen, and P. Shi, "An algorithm for license plate recognition applied to intelligent transportation system," *IEEE Trans. Intell. Transp. Syst.*, vol. 12, no. 3, pp. 830–845, Sep. 2011.

[36] I. Mayire and H. Askar, "Design and implementation of prototype system for online handwritten Uyghur character recognition," *Wuhan Univ. J. Natural Sci.*, vol. 17, no. 2, pp. 131–136, Apr. 2012.

[37] J. Zeng and Z. Q. Liu, "Markov random field-based statistical character structure modeling for handwritten Chinese character recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 5, pp. 767–780, May 2008.

[38] N. Arica and F. T. Yarman-Vural, "Optical character recognition for cursive handwriting," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 6, pp. 801–813, Jun. 2002.

[39] G. A. Montazer, H. Q. Saremi, and V. Khatibi, "A neuro-fuzzy inference engine for Farsi numeral characters recognition," *Expert Syst. Appl.*, vol. 37, no. 9, pp. 6327–6337, Sep. 2010.

[40] B. Zhu and M. Nakagawa, "A robust method for coarse classifier construction from a large number of basic recognizers for on-line handwritten Chinese/Japanese character recognition," *Pattern Recognit.*, vol. 47, no. 2, pp. 685–693, Feb. 2014.

[41] N. Das, R. Sarkar, S. Basu, P. K. Saha, M. Kundu, and M. Nasipuri, "Handwritten *Bangla* character recognition using a soft computing paradigm embedded in two pass approach," *Pattern Recognit.*, vol. 48, no. 6, pp. 2054–2071, Jun. 2015.

[42] G. Vamvakas, B. Gatos, and S. J. Perantonis, "Handwritten character recognition through two-stage foreground sub-sampling," *Pattern Recognit.*, vol. 43, no. 8, pp. 2807–2816, Aug. 2010.

[43] K. Verma and R. K. Sharma, "Comparison of HMM- and SVM-based stroke classifiers for Gurmukhi script," *Neural Comput. Appl.*, vol. 28, pp. 51–63, Dec. 2016.

[44] A. A. Desai, "Gujarati handwritten numeral optical character reorganization through neural network," *Pattern Recognit.*, vol. 43, no. 7, pp. 2582–2589, Jul. 2010.

[45] D. Khanduja, N. Nain, and S. Panwar, "A hybrid feature extraction algorithm for devanagari script," *ACM Trans. Asian Low-Resour. Lang. Inf. Process.*, vol. 15, no. 1, pp. 2:1–2:10, Jan. 2016.
[46] N. R. Soora and P. S. Deshpande, "Novel geometrical shape feature extraction techniques for multilingual character recognition," *IETE Tech. Rev.*, vol. 34, no. 6, pp. 612–621, Oct. 2016.
[47] M. Kumar, R. K. Sharma, and M. K. Jindal, "Efficient feature extraction techniques for offline handwritten Gurmukhi character recognition," *Nat. Acad. Sci. Lett.*, vol. 37, no. 4, pp. 381–391, Aug. 2014.
[48] M. Kumar, M. K. Jindal, and R. K. Sharma, "A novel hierarchical technique for offline handwritten Gurmukhi character recognition," *Nat. Acad. Sci. Lett.*, vol. 37, no. 6, pp. 567–572, Dec. 2014.
[49] M. Kumar, R. K. Sharma, and M. K. Jindal, "A novel feature extraction technique for offline handwritten *Gurmukhi* character recognition," *IETE J. Res.*, vol. 59, no. 6, pp. 687–691, Sep. 2014.
[50] S. Bag, G. Harit, and P. Bhowmick, "Recognition of Bangla compound characters using structural decomposition," *Pattern Recognit.*, vol. 47, no. 3, pp. 1187–1201, Mar. 2014.
[51] N. R. Soora and P. S. Deshpande, "Robust feature extraction technique for license plate characters recognition," *IETE J. Res.*, vol. 61, no. 1, pp. 72–79, Jan. 2015.
[52] I. Bazzi, R. Schwartz, and J. Makhoul, "An omnifont open-vocabulary OCR system for English and Arabic," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 21, no. 6, pp. 495–504, Jun. 1999.
[53] M. Mori, S. Uchida, and H. Sakano, "Global feature for online character recognition," *Pattern. Recogn. Lett.*, vol. 35, pp. 142–148, Jan. 2014.
[54] U. Pal and N. Tripathy, "A contour distance-based approach for multi-oriented and multi-sized character recognition," *Sadhana*, vol. 34, no. 5, pp. 755–765, Oct. 2009.
[55] K. C. Santosh and L. Wendling, "Character recognition based on non-linear multi-projection profiles measure," *Frontiers Comput. Sci.*, vol. 9, no. 5, pp. 678–690, Oct. 2015.
[56] R. Manmatha and J. L. Rothfeder, "A scale space approach for automatically segmenting words from historical handwritten documents," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 8, pp. 1212–1225, Aug. 2005.
[57] J.-M. Geusebroek, A. W. M. Smeulders, and J. van de Weijer, "Fast anisotropic Gauss filtering," *IEEE Trans. Image Process.*, vol. 12, no. 8, pp. 938–943, Aug. 2003.
[58] V. N. Vapnik, *Statistical Learning Theory*. Hoboken, NJ, USA: Wiley, 1998.
[59] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proc. 5th Annu. Workshop Comput. Learn. Theory (COLT)*, Pittsburgh, PA, USA, 1992, pp. 144–152.
[60] D. Lewis, G. Agam, S. Argamon, O. Frieder, D. Grossman, and J. Heard, "Building a test collection for complex document information processing," in *Proc. 29th Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Seattle, WA, USA, 2006, pp. 665–666.
[61] G. Agam, S. Argamon, O. Frieder, D. Grossman, and D. Lewis. (2006). *The Complex Document Image Processing (CDIP) Test Collection Project*, Illinois Institute of Technology. [Online]. Available: http://www.umiacs.umd.edu/~zhugy/tobacco800.html
[62] University of California, San Francisco. (2007). *The Legacy Tobacco Document Library (LTDL)*. [Online]. Available: http://legacy.library.ucsf.edu/
[63] T. E. de Campos, B. R. Babu, and M. Varma, "Character recognition in natural images," in *Proc. Int. Conf. Comput. Vis. Theory Appl.*, Lisboa, Portugal, 2009, pp. 1–8.
[64] M. Diem, S. Fiel, A. Garz, M. Keglevic, F. Kleber, and R. Sablatnig, "ICDAR 2013 competition on handwritten digit recognition (HDRC 2013)," in *Proc. Int. Conf. Document Anal. Recognit.*, Washington, DC, USA, Aug. 2013, pp. 1422–1427.
[65] R. Kumar, A. Kumar, and P. Ahmed, *A Benchmark Dataset for Devnagari Document Recognition Research*. Lemesos, Cyprus: WSEAS Press, 2013.
[66] V. J. Dongre and V. H. Mankar, "Development of comprehensive Devnagari numeral and character database for offline handwritten character recognition," *Appl. Comput. Intell. Soft Comput.*, vol. 2012, May 2012, Art. no. 871834. [Online]. Available: http://dx.doi.org/10.1155/2012/871834
[67] V. J. Dongre and V. H. Mankar, "Devnagari handwritten numeral recognition using geometric features and statistical combination classifier," *Int. J. Comput. Sci. Eng.*, vol. 5, no. 10, pp. 856–863, Oct. 2013.

PARUL SAHARE received the M.Tech. degree in VLSI design from the Visvesvaraya National Institute of Technology, Nagpur, India, in 2010, where he is currently pursuing the Ph.D. degree. He has a total of three years of academic experience. His area of interest includes signal processing, image processing, and pattern recognition.

SANJAY B. DHOK received the Ph.D. degree in electronics engineering from the Visvesvaraya National Institute of Technology, Nagpur, India. He is currently an Associate Professor with the Centre for VLSI & Nanotechnology, Visvesvaraya National Institute of Technology. He has authored many research papers in national and international journals and conferences. His area of interest includes signal processing, image processing, data compression, wireless sensor networks, and VLSI design. He is a member of the IEEE Society.