Hindawi
Scientific Programming
Volume 2020, Article ID 8866041, 12 pages
https://doi.org/10.1155/2020/8866041

Research Article

An Efficient Skewed Line Segmentation Technique for Cursive Script OCR

Saud Malik,[1] Ahthasham Sajid,[2] Arshad Ahmad,[3] Ahmad Almogren,[4] Bashir Hayat,[5] Muhammad Awais,[6] and Kyong Hoon Kim[7]

[1]*COMSATS University Islamabad, Attock Campus, Islamabad 43600, Pakistan*
[2]*Department of Computer Science, Faculty of ICT, BUITEMS, Quetta, Baluchistan 87300, Pakistan*
[3]*Department of IT and Computer Science, Pak-Austria Fachhochschule: Institute of Applied Sciences & Technology, Khanpur Road, Mang, Haripur 22620, Pakistan*
[4]*Department of Computer Science, College of Computer & Information Sciences, King Saud University, Riyadh 11633, Saudi Arabia*
[5]*Institute of Management Sciences, Peshawar 25000, Pakistan*
[6]*School of Computing and Communications, Lancaster University, Bailrigg, Lancaster LA1 4YW, UK*
[7]*School of Computer Science & Engineering, Kyungpook National University, Daegu 41566, Republic of Korea*

Correspondence should be addressed to Arshad Ahmad; yaarshad@gmail.com

Received 27 August 2020; Revised 4 November 2020; Accepted 8 November 2020; Published 4 December 2020

Academic Editor: Shaukat Ali

Segmentation of cursive text remains the challenging phase in the recognition of text. In OCR systems, the recognition accuracy of text is directly dependent on the quality of segmentation. In cursive text OCR systems, the segmentation of handwritten Urdu language text is a complex task because of the context sensitivity and diagonality of the text. This paper presents a line segmentation algorithm for Urdu handwritten and printed text and subsequently to ligatures. In the proposed technique, the counting pixel approach is employed for modified header and baseline detection, in which the system first removes the skewness of the text page, and then the page is converted into lines and ligatures. The algorithm is evaluated on manually generated Urdu printed and handwritten dataset. The proposed algorithm is tested separately on handwritten and printed text, showing 96.7% and 98.3% line accuracy, respectively. Furthermore, the proposed line segmentation algorithm correctly extracts the lines when tested on Arabic text.

1. Introduction

The OCR systems have standard measures, different types, and rich history. Urdu language is widely spoken and understood in mainly South Asian countries. In contrast to its vast usage, little to no improvement has been made for recognition of its script [1]. The script recognition system for Urdu printed and handwritten text has been approached recently as compared to the OCR systems for other scripts. This void of research is mainly due to the lack of benchmark datasets, dictionaries, and other necessary factors.

In recent years, the OCR system has attracted a lot of researchers' attention towards cursive text scripts, while the text of languages like Urdu and Arabic is still far behind in attaining convincing accuracy [2]. The writing pattern of any language is the main reason for the high or low accuracy rate of text recognition. In this context, cursive text (Arabic, Urdu, etc.) is having more accuracy concerns. Line segmentation and recognition of line in the cursive script is a hard task because of their words shape and composition. Some of the cursive scripts are mentioned in Table 1. OCRs of cursive languages are not so mature that there is still room for improving accuracy. Urdu handwritten OCR is very beneficial, but we cannot attain its benefits until we overcome problems in segmentation.

In the field of pattern recognition and image processing, English text OCR systems attain good accuracy. In the OCR system, segmentation is the most error generating

Table 1: Different cursive scripts.

The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog.
English cursive script (printed)
The quick brown fox jumps over the lazy dog.
English cursive script (handwritten)
یہ گھر کس کا ہے
Arabic Naskh script
یہ گھر کس کا ہے
Urdu Nastalik script

part in cursive scripts like Urdu and Arabic. That is why segmentation-free approaches are used to handle this problem to some extent. But in large datasets, this approach also failed to give satisfactory accuracy. As an alternative, we have segmentation-based approaches.

Besides the aforementioned motivations, the aim of this study is also to enhance the segmentation accuracy and progress in the state-of-the-art methodologies. The major objective is to solve segmentation accuracy for Urdu OCR, starting from text line segmentation up to ligature segmentation. In cursive scripts, the skewness of a text image is also affecting the segmentation stage. Segmentation of overlapped and skewed text lines is a hurdle in attaining high segmentation accuracy [3]. So, in this research work, skew detection and baseline detection are also taken to uplift the segmentation method. Finally, the ligature segmentation approach is also discussed to fully cover the segmentation stage and achieve high recognition accuracy through the proposed method. In cursive scripts, beside character segmentation, line segmentation is also the error generating part because of the following problems.

Overlapped text lines: in cursive scripts, words of adjacent lines overlap each other. This problem occurs in both handwritten and printed text. Some words are connected with words of adjacent lines and partly segmented during the segmentation process. Due to this, there is a high probability of loss of information.

Unequal line height: this issue occurs in only handwritten text. As there is no standard writing throughout the document, the height of words varies in line, there is no uniform position of words, and characters spread in both vertical and horizontal directions; because of this, a line may not remain straight.

Inconsistent space between lines: in Urdu's handwritten text, there is no standard baseline on which text is written. The space between text lines varies throughout the document. In such cases, baseline is not horizontally straight; it may be in a curve or oscillatory shape. For example, in some cases, at the beginning, start space between lines is minor and at the end of adjacent lines, there is significant space between them which causes skewness problems in segmenting text lines.

Dot/diacritics overlapping: in Urdu, dot/diacritics are mostly spread in white spaces between lines. In these cases, always there is a chance that a dot/diacritic of a line may segment with the adjacent line.

This research contributed to the following directions:

(i) Firstly, a printed and handwritten Urdu text dataset is generated along with ground truth (by removing the skewness of the next page). Afterward, the page is converted to lines and ligatures.

(ii) Secondly, the proposed scheme is evaluated on a manually generated Urdu printed and handwritten dataset, which consists of 80 Urdu handwritten pages (687 lines) and 48 printed pages (495 lines).

(iii) Thirdly, the presented framework addressed existing problems in text line segmentation which are variable text size, the inconsistency of gap between lines, and skewness of text lines.

(iv) In the end, the proposed algorithm is tested separately on handwritten and printed text, showing 96.7% and 98.3% line accuracy, respectively.

Results validated that the proposed line segmentation algorithm correctly extracts the lines when tested on Arabic text.

Till now, we have discussed the OCR system, different types of text, and their problem. Section 2 consists of the related work of the study. Section 3 contains the proposed methods of the study. Then, the results and discussion are addressed in Section 4, and finally, the paper is concluded in Section 5.

2. Related Work

The modes for an OCR system can be categorized according to input text image acquisition that can be offline or online, writing mode that can be printed or handwritten, and finally the font variations leading to font constraints, handled by a recognition system with the support of segmentation process. In this section, we will review existing techniques for text segmentation of printed and handwritten cursive scripts.

Projection profile is the widely used segmentation approach for line segmentation. This method automatically identifies line regions from an image. In this technique, information is used for text line segmentation, while this technique does not give good results for an image having inconsistent line spacing and skewness also affects the accuracy. The skewed image develops small peaks which make the projection profile confused about proper text lines; this problem is solved by using local or piece-wise projection profiles [4]. A modified projection profile is used as an adaptive technique in which partial line fractions are used to develop projection [5]. In [6], projection profile is used to segment ligatures during the line segmentation stage while Y-histogram projection is used for line segmentation. The projection profile based approach is also employed in [7] for baseline detection and division of connected components and then words are segmented into ligatures through vertical

projection. Our approach also used the vertical projection technique for ligature segmentation from lines.

Muna Ayesh et al. [8] presented an algorithm to segment the Arabic text lines. In Arabic, line segmentation problem is mainly due to the placement of diacritics. As diacritics do not follow any baseline, they merge with neighboring lines. The authors presented a line and diacritics segmentation algorithm tested on 43000 lines, giving 99.5% accuracy. The projection profile approach is used with a profile amplitude filter in [9] for Arabic text line, words, and character segmentation. This algorithm is for printed documents irrespective of word size and fonts. Line segmentation algorithms work in two steps. Firstly, rough segmentation is done using a horizontal profile method; after that, the proposed technique having multiple rules is applied to rough segmentation to get final segmentation results.

In [2], the authors presented a segmentation approach using chain code. In this approach, the start point is detected after thinning. Through Freeman chain code, segmented points are marked, while the HMM model is used to recognize each segment as a character or diacritic. Line segmentation (overlapping) and ligature segmentation (primary and secondary) algorithms are presented in [10]. The hybrid top-down approach (projection profile) is used for line segmentation, while the bottom-up approach is for ligature segmentation, in which connected components are extracted and collected for ligature extraction. After that, diacritics and ligatures are categorized as primary and secondary components. This technique gives an accuracy of 99.02% in ligature segmentation and 99.11% in line segmentation.

Moysset et al. [11] used one of the recent models of recurrent neural network for line segmentation. A six-layer deep neural network is used, in which four LSTM layers and the remaining two are a convolutional layer. This approach addresses the location of the text line. Novel line segmentation technique is used in which information energy of pixels is used to segment text lines. The characters are also segmented by using an artificial neural network. The method gives 95% accuracy for line segmentation and 94% for character segmentation.

To segment palm leaf manuscripts of Dai, Ge Peng et al. [12] used algorithm based on HMM, to evaluate all segmentation paths. In another study, Quang Nhat and Lee [13] proposed multilingual text line segmentation approach, in which trained fully conventional network (FCN) is used to figure out text lines pattern. Through FCN, line map is extracted through which initial segmentation is done and after that line, adjacency graph is used to handle overlapping words between lines. This gives 98.6% accuracy on ICDAR-2013.

In water flow approaches, angles of a document (left-right and top-bottom) are used for hypothetical water flow. This algorithm works on this hypothetically assumed situation. For line extraction, strip of un-wetted areas is used. It is assumed that spaces between lines are to be filled and form wet areas after labeling images divided into two parts, wetted and non-wetted areas, where the wetted areas contain spaces and the remaining un-wetted areas contain text lines. Darko Brodi'c [14] modified a linear water flow algorithm, by changing its linear function by power function. Waterflow deals with linearly straight lines; in this modified algorithm, bounding boxes are added to handle angular dimensions of the text.

The smearing approach is considered as one of the earliest approaches used for line segmentation. The smearing approaches refer to the concept of smearing all consecutive dark pixels along the horizontal direction between lines. Then, white spaces within dark pixels are filled with black pixels. Through this, black pixels cover a large area along with the text. This black pixel growing area enclosed a separated text line [15]. In recent times, the RLSA algorithm has been introduced which is based on smearing techniques. Novel painting approach is presented to smear the foreground portion of the image; this way, foreground is separated from background pixels this method is used for text line segmentation.

Probabilistic algorithms are used in stochastic approaches. This method accomplishes the nonlinear path in between overlapping text lines. Then, HMM is used to extract these text lines and the image is divided into small units. In the case of touching components, a high probability path crosses the touching component with minimum dark pixels, while this method drops accuracy in text having a large number of the black pixels contact point of text [16, 17].

Kumar and Choudhary [18] proposed an algorithm for English cursive script. This algorithm vertically segments the ligatures of connected words. First, get the single-pixel stroke width of the scanned image by skew angle correction and thinning. The algorithm segments characters on the base of their geometrical shape. The proposed method is tested on the local dataset. In [19], researchers compared different segmentation algorithms. Experiments are carried out on the CCC benchmark dataset. The horizontal and vertical projection method gives 95.65% segmentation accuracy while Hough transform technique shows 98.9% accurate line segmentation.

In [20], Naz et al. used implicit segmentation for Urdu text segmentation. The horizontal projection profile is integrated into the segment page into lines. Different features of the text, like zone based statistical measures and chain codes, etc., are computed and then neural network is trained for recognition. The method is evaluated on the UPTI dataset. Line and ligature segmentation algorithms are presented in [21].

In [22], the author presents a set of rules that are derived heuristically to search character boundaries of the cursive script that is validated by using an ensemble of neural confidence. Rehman [23] introduced a new concept of core-zone for segmenting such difficult slanted handwritten words. Also in [24], Qaroush et al. used the baseline detection method for the segmentation of characters and CNN for recognition of characters.

Mullick et al. [25] presented a novel approach to segment lines from handwritten Bangla document image. In this approach, first blur the white spaces between words (so that after blurring the white spaces in between words, only the remaining white spaces are in between lines). This way, the most prominent pixels that remained are points of separation.

In [26], multilingual novel baseline approach is used for handwritten text line segmentation, in which significant contours are extracted for making the curve. Orientation invariant features of this curve are used to determine whether the extracted region is a baseline of the text line or not. SVM is trained using the orientation invariant features of the curves and then trained SVM is used to figure outlines from the text. This approach gives 89.6% baseline accuracy.

In [27], Surinta et al. proposed an algorithm to segment lines from historical documents. The novelty of this approach is its artificial agent which handles the abnormality of historical text. In this approach, both ends of each line are detected using the smoothed horizontal ink density histogram. The proposed algorithm uses different cost functions to keep a distance from ink pixels and computes the shortest distance between them to detect lines. This method gives 99.9% line segmentation accuracy on the Saint Gall dataset.

3. Proposed Methods

For Urdu cursive script text, in recent decades a lot of research has been carried out for Urdu printed (Nastaliq) script. But in the case of Urdu handwritten text, a lot of efforts are needed to develop an algorithm that leads to an ideal OCR system. This paper proposes a method for skew correction and line segmentation of printed and handwritten documents. The presented methodology works in the following sequential steps:

(1) Preprocessing

(2) Skew correction

(3) Text line segmentation

(4) Ligature\word segmentation

The focus of this paper is mainly on preprocessing and segmentation of Arabic/Urdu scripted OCR. Before segmenting text into its basic shape, the system preprocesses the noisy and skewed images. This paper reflects a segmentation technique along with the projection profile technique. The paper reflects an enhanced pixel counting based robust algorithm which is independent of any script-specific knowledge. Moreover, a modified header and baseline detection technique [28] is used for the line segmentation algorithm.

Pixel based method mainly lacks in detecting noisy and skew text images. We overcome both the drawbacks of the header and baseline method in this proposed method by employing the adaptive threshold method for noise detection and a novel algorithm for skew detection. The graphical representation of the method is shown in Figure 1.

3.1. Preprocessing. Preprocessing is a very essential phase for better segmentation results. For preprocessing, adaptive thresholding approach is used, in which noisy image is given as input. Adaptive thresholding is applied to the input image which is based on the intensity of the image. Grayscale image contains image pixel intensity. For the RGB image, the image is converted into YCbCr color space, where Y contains (black and white pixels) intensity, 8-bit depth (grayscale), and 24-bit depth (RGB) images can be used in this method. After converting the image into grayscale, adaptive thresholding is applied.(Algorithm 1).

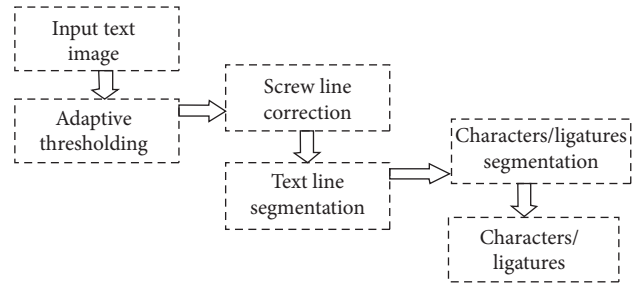


Figure 1: Proposed methodology.

An input image is given as grayscale or RGB format, for processing convert it into a binary image (white and black image). Text image has mainly two colors: (1) text color and (2) background color. By converting into binary images, the contrast of the image increases and is easy for global thresholding using Otsu's method [29].

An adaptive local thresholding algorithm separates the background from the foreground of the image with nonuniform brightness. Apply adaptive (mean/median) filter to highlight image features and then apply the Otsu threshold to generate a binary image. It has been found that Otsu's thresholding has produced good results on text data. For better-preprocessing results, the binarization technique is used, in which pixels having more than specified intensity will be converted into white pixels; otherwise, they will be converted into black pixels; therefore, the image is converted into black and white pixels. The equation is as follows:

$$\sigma_T^2 = \sum_{i=0}^{L-1} (i - \mu T)^2 Pi, \quad (1)$$

where P is probability and I is the intensity of L number of bins. After that, a single-pixel thick edge boundary (in the whole image) is created for each character from a canny edge detector (auto-thresholding). As per our knowledge, the canny edge detector is used for preprocessing, as it removes dot noise with a low-pass filter. At that point, apply a Sobel filter and then use nonmaximal suppression to choose the finest pixel for edges when there are different local possibilities.

3.2. Skew Detection. The proposed line segmentation is dependent on Algorithm 2, skew correction algorithm. A line segmentation algorithm requires skewing fewer images for good performance. The proposed skew correction algorithm works on the pixels intensity information of the image. The main idea of this algorithm is to extract the area between text lines and fit it to a straight line.

Find the white spaces (in the case if the background is white) that define the zones between the lines with letters on them. Rather than finding the text, we will find the white spaces between the lines. Then, we will go from the center of the first spaceline to the center of the next one, till the end of the page. In this method, skew correction is considered at the page level. At last, find the tilted angle of lines and rotate the image around its center point.

Input: Grayscale image or RGB.
Output: Clean image.
//Begin
Step 1. //If the image is RGB, convert it into YCbCr color space.
YCbCr ← RGB
Step 2. //Eliminate CBCR. And the image becomes grayscale.
Step 3. //Apply adaptive thresholding.
Step 4. Apply the global image threshold using Otsu's method.
Step 5. Adaptive (mean\median) filter to highlight image features.
Step 6. Then, apply the Otsu threshold to segment and generate a binary image.
//End

Algorithm 1: Preprocessing.

Input: Noiseless skewed image.
Output: Skew-less image.
//Begin
Step 1. Convert input text image into a grayscale image.
Step 2. Scan the document and extract ROIs. // ROI = area between text lines.
Step 3. Find all pixels between text lines (other than text pixels).
Step 4. Join these pixels to fit on a line (having the same slope as text).
Step 5. Go for the center of each fitted line.
Step 6. Find the angle between the fitted line and the horizontal line of the page.
Step 7. Rotate area.
//End

Algorithm 2: Skew correction algorithm.

3.3. Line Segmentation. Noiseless and skew-less images are fed to a system for further processing. The input binarized image is inverted for getting an image I'HXW with black text on a white background. Then, pixel strength (P) is calculated for the black text in a document.

$$P = \sum_{y=1}^{W} l'(x, y)[1, H]. \tag{2}$$

This pixel strength (P) determines the threshold of text image; it varies from image to image. Rows having dark pixels greater than the standard deviation value of the document are extracted. The text line is the combination of consecutive rows, which are extracted as pixels (P) lying between header and footer.

$$\begin{aligned} &\text{For header}: \ Px_i: \ x_{i-1} < x_i > x_{i+1}, \\ &\text{For baseline}: \ Px_l': \ x_{i-1}' > x_i' < x_{i+1}'. \end{aligned} \tag{3}$$

The above equations represent the criteria for assigning header and baseline to a particular text line. Two Lines in a text are separated when black pixels in a row are less than the adaptive threshold in rows of the entire document, shown by "white spaces" in Figure 2. This white pixel acts as a border between two text lines.

As this technique focuses on the start and end of a text line, this is why it is referred to as the "header and baseline detection" technique. The text line is extracted by using two parameters of the text line, the starting point of the line and the baseline (last row of the line). Header and baseline are determined by the number of black and white pixels in rows of text images. Exceeding threshold consecutive black pixel rows are labeled as text line whereas repeated white pixel rows are considered as the separation area between two lines (shown in Figure 2). Pictorial representation of the line segmentation algorithm is shown in Figure 3.

In the proposed method, the adaptive threshold of the page is set by calculating the standard deviation of text pixels (black for white background), which determines the diversity of text pixels on a page. It works on the idea that higher value of standard deviation means greater distance between lines. This adaptive threshold determines the minimum number of consecutive text rows in a line. As in the case of Arabic/Urdu scripted text, diacritics of text appear above the line and contain fewer pixels. In some cases, those rows which have black pixels less than the minimum threshold value affect recognizing text pixels by not detecting dot/diacritics in a line. Algorithm 3 addresses the proposed methodology with all abbreviations in Table 2.

This technique purely depends on the counting pixels approach. The main idea of this technique is that lines of page contain larger number of black pixels than the spaces between lines. Height threshold is used for the height of segmented line. Threshold is set according to the page in consideration. If the spaces between the lines are constant, then the algorithm is tuned to fix the threshold value according to spaces between lines, while if the line distance

between the adjacent lines is changing constantly, then the algorithm must be capable of adaptive thresholding.

For ligature segmentation, the projection profile method is used for text line segmentation and lines are transformed into words through the vertical profile method. In the proposed method, Urdu handwrote, or printed text page, is inserted as an input. Firstly, the page is segmented into several text lines and further fed into a word/ligature segmentation algorithm which divided lines into the smallest possible ligatures. As the algorithm segments words in sequential order, ligatures are automatically arranged in sequence.

3.4. Dataset Generation. Urdu's handwritten dataset is composed of a collaboration of 24 writers. Each participant writes a different number of words, lines, and pages having 687 lines which combine to form 80 pages. The number of words in a line and the number of lines on a page vary throughout the dataset. Sample data is versatile with almost all types of writing problems that make the dataset complex and reduce the accuracy of the algorithm; for example, each writer has their own writing style having different difficulty levels of recognition, and each participant is having a different text size. These handwritten text images were captured using a high-resolution digital camera and stored in jpg format after scanning. Samples of handwritten dataset are shown in Figure 4. Details of the dataset are presented in Table 3 and available online on GitHub (https://github.com/saud00/Urdu-text-dataset).

To evaluate the algorithm on Urdu OCR, we present a diverse and comprehensive Urdu handwritten dataset. The dataset is written with a blue and black ball pen and also pointer pen is used so that the dataset contains all types of writing intensity. Then, this dataset is scanned and converted to a binary image (white and black). Eighteen teachers and ten students (both male and female) contributed to the dataset. They were told to write paragraphs (without any restriction of content). They wrote different numbers of lines in different writing with different pens. Samples are shown in Figure 4. The ground truth is also manually created for a handwritten dataset. They were not trained so that this database reflects the true essence of challenging real databases.

Urdu Nastaliq printed dataset is also generated and the data is collected from three different sources.

(i) 27 pages are collected from online books (by taking a screenshot and then cropping through Paint)

(ii) 10 pages of newspapers (scanned from a camera)

(iii) 11 pages of digests (scanned from a camera)

To maintain diversity in the dataset, data is collected from three different sources. Firstly, 27 pages of the online book (Shahab Nama) are collected (Figure 4(c)), by taking screenshots of twenty-seven pages and then cropping them through Paint. Twenty-seven pages contain a total of 275 lines. Secondly, ten paragraphs of newspapers are collected through a digital camera and then scanned for further use (Figure 4(a)). Ten paragraphs of newspapers contain 86 lines. Finally, randomly 10 pages are scanned from a digest (Figure 5(b)), which contains 131 lines.

4. Results Analysis and Discussion

The above-generated dataset is tested for evaluating the proposed method. It is ensured in the dataset that it contains images of possibly all renowned formats: jpg, png, and grayscale, etc. For compiling results, MATLAB 2017a is used. The code of this project is available on GitHub (https://github.com/saud00/Line_and_Word_Segmentation_URDU) along with dataset and output images.

The accuracy of the proposed line segmentation is dependent on the accuracy of skew correction. Line segmentation algorithm requires skew-less image for good performance. The proposed skew algorithm works on the pixels intensity information of image.

Results are generated at the end of segmentation stage, later used for recognition rate. Firstly, preprocessing technique is applied to remove noise so as to handle false line detection; after that, if the image is tilted in either way, it is removed by applying skew detection algorithm. De-skewed image is fed into line segmentation algorithm for segmenting lines.

4.1. Results Analysis. The accuracy of the proposed framework is tested with labeled ground truth text line images. The dataset contains 495 printed and 681 handwritten line images snapshots along with manually created ground truth. As part of a framework, the dataset is developed and labeled manually. The number of characters in labeled ligature images is counted and compared with several recognized characters to find recognition accuracy using

$$\text{accuracy} = \frac{\sum_{i=1}^{n} L_i - \left(\sum_{i=1}^{n} L_i - \sum_{i=1}^{n} R_i\right)}{\sum_{i=1}^{n} L_i} \times 100, \tag{4}$$

where R represents correctly recognized lines and L is input labeled line image. The segmented lines from the input page are compared with ground truth images to find line recognition accuracy. For evaluating results, we use precision, recall, and F-measure matrices which are defined as follows:

$$\begin{aligned} \text{recall} &= \frac{\text{correctly segmented lines by the algorithm}}{\text{actual total number of lines}}, \\ \text{precision} &= \frac{\text{correctly segmented lines by algorithm}}{\text{total number of lines segmented by the algorithm}}, \\ \text{F} - \text{measure} &= \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}. \end{aligned} \tag{5}$$

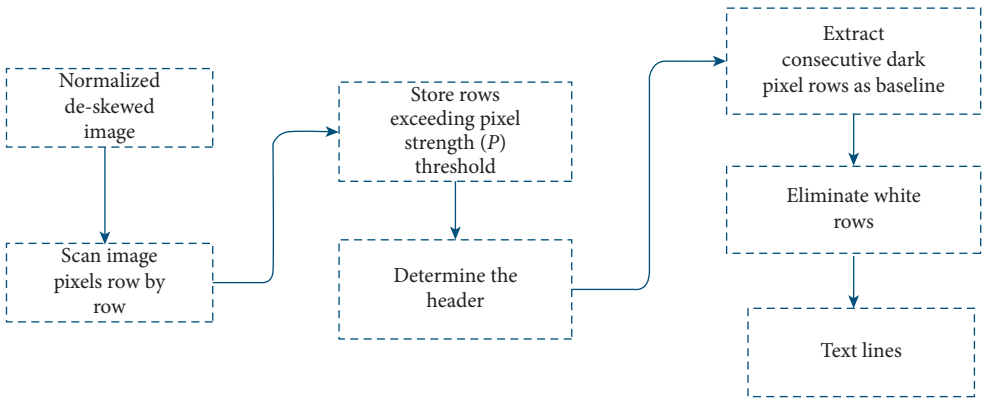


Figure 2: Header and baseline in the text image.

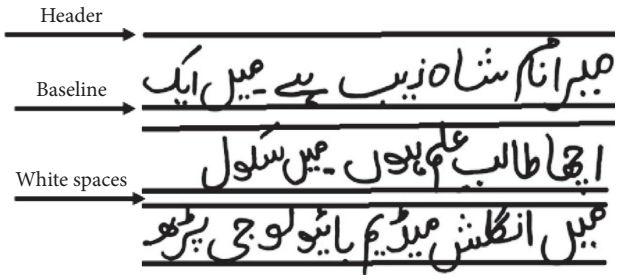


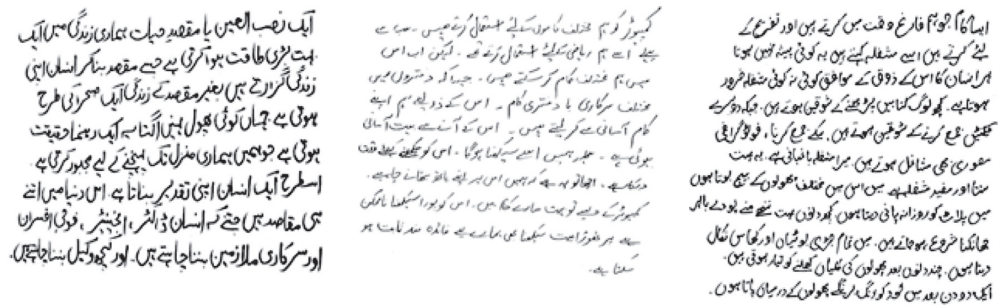
Figure 3: Flow chart of line segmentation methodology.

Input: Normalized de-skewed image
Output: Segmented lines.
//Begin.
Step 1. //Preprocessing: image binarization (using adaptive threshold).
Step 2. //De-skew the image (if needed).
Step 3. //Scan Image row by row.
Identify the intensity for each pixel (0 or 1).
Step 4. //Calculate the standard deviation of the image (use as minimum black pixels in a text row).
Step 5. If(Black_Pixels > Std)
Black_Row = row
Step 6. Else
Space_Row = row
Step 7. For (Start from 1st_Black_Row: till Last_space_Row)
Step 8. If (Height_Row > Min_ Height_Row)
//Consider these consecutive text rows as text line until any white_row occurs.
Step 9. If (Space_Row occur)
Step 10. If (Space_Row > Min_Height_Row)
Break text_line and go to Step 11.
Step 11. else
Go to Step 7
Step 12. else
Search for next black_text_row
Step 13. Else
Go to Step 10
//End

Algorithm 3: Text line segmentation algorithm.

Table 2: Abbreviations used in Algorithm 3.

Abbreviations	Description
Std	The standard deviation of the image
Black_Pixels	Number of black pixels in a row
Black_Row	Row having black text pixels greater than the threshold
Space_Row	Space between text lines
Space_Row	Space between text lines
1st_Black_Row	Last space_row of page
Height_Row	Number of consecutive black_text_rows
Min_ Height_Row	Minimum threshold of consecutive black rows



(a) (b) (c)

Figure 4: Handwritten dataset samples.

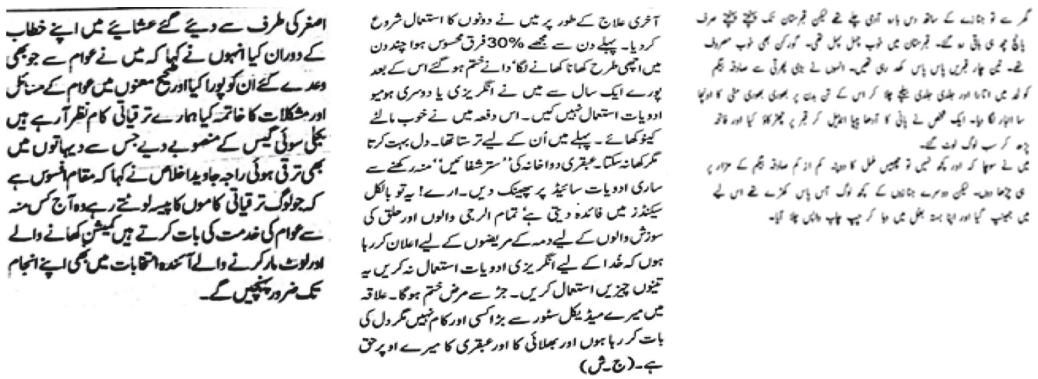
Table 3: Details of handwritten dataset.

Dataset details	Statistics
Total number of handwritten pages	80
Total number of writers	28
Number of pages written by a writer	3 (average)
Number of text lines per page	9 text lines (average)
Number of skewed pages	12
Total number of lines	687
Number of skewed lines	97
Approx. number of words per page	103
Approx. number of lines by a writer	27
Approx. number of lines by a writer	306
Approx. number of words per line	12
Total number of words	8,208

Precision is the fraction of relevant segmented lines among the retrieved lines, while recall is the fraction of relevant lines that have been retrieved over the total amount of relevant lines. The skew detection algorithm detects the skew of the text image by using the proposed skew correction algorithm. Figure 6(b) shows the output of the skew correction algorithm. The skew correction algorithm is evaluated based on true line segmentation. A total of 13 skewed images are tested on the algorithm, from which 11 images are truly de-skewed.

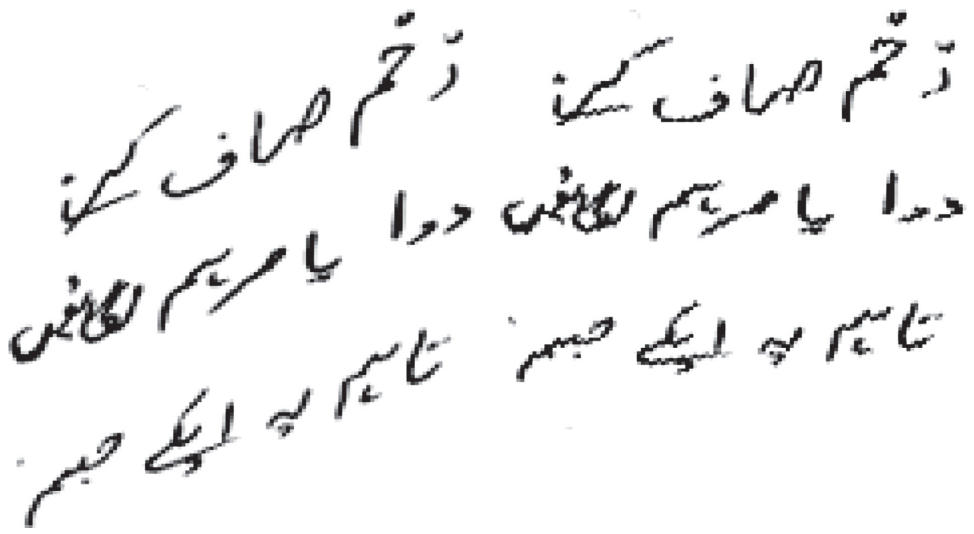
4.1.1. Handwritten Documents. The framework is tested on both handwritten and printed Urdu text. Dataset of 80 pages (687 lines) is evaluated on the text line segmentation algorithm. Line segmentation algorithm correctly segments 687 lines, from which 8 lines are under-segmented and incorrect 18 lines are formed while over-segmentation issue affects 4 text lines. Line segmentation algorithm gives 96.7% line accuracy by correctly spotting 665 handwritten text lines. The result is shown in Table 4.

4.1.2. Printed Documents. The algorithm is tested on the Urdu Nastaliq printed dataset which is collected from 3 different mediums online books, newspapers, and digest. From a total of 48 pages, 27 pages are taken from an online book, 10 pages from a newspaper, and 11 scanned pages collected from digest shown in Table 5. For printed data, a total of 48 pages (495 lines) are tested. The result of the printed dataset is given in Table 5 which shows 98.38% accuracy by detecting 487 lines from a total of 495 lines.



(a) (b) (c)

Figure 5: Printed dataset samples. (a) Newspaper sample. (b) Digest sample. (c) Online page sample.



(a) (b)

Figure 6: (a) Skewed image. (b) Output image.

Table 4: Evaluation matrices.

Text type	No. of lines	Detected lines	Correctly detected lines	Precision	Recall	F-measure
Handwritten	687	681	665	97.6	96.79	97.3
Printed	495	491	487	99.18	98.38	98.74

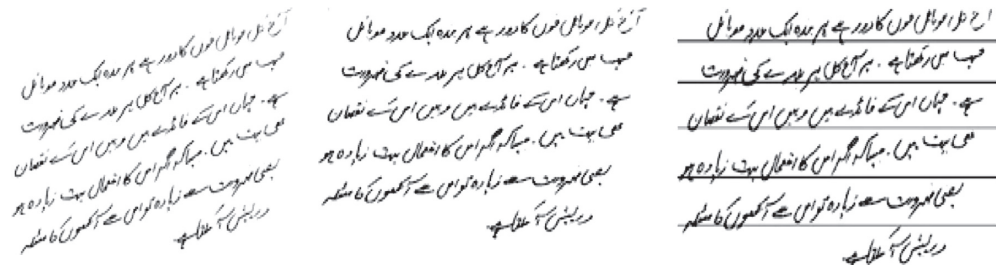
Table 5: Details of Urdu printed dataset and its corresponding results.

Source	Pages	Lines	Correctly detected lines	Accuracy
Online book	27	275	272	98.9
Newspaper	10	86	84	97.7
Digest	11	131	131	100
Overall	48	495	487	98.38

Among 3 types of data type, the algorithm detects lines of digest with ease by showing 100% accuracy. We fed 11 pages of digest having 131 lines; these all are correctly recognized. From 275 lines of an online book, only 3 misled the proficiency, showing a 98.9% recognition rate.

From the overall 48 printed pages, the newspaper shares 11 pages with 86 lines. The algorithm detects 84 lines. But, there are a lot of differences between paragraphs of the newspaper (Figure 5(a)) and paragraphs of digest (Figure 5(b)).

The number of pages in the newspaper and digest is nearly the same but having a large difference in the number of lines. The algorithm correctly detects all the 131 lines in digest and attains 100% accuracy. Figure 7 shows the whole



(a) (b) (c)

Figure 7: (a) Original image, (b) de-skewed image, (c) output of proposed line segmentation algorithm.

(a)

(b)

Figure 8: (a) Original image, (b) ligature segmented image.

Table 6: Comparison of the proposed technique with the existing methods.

Source	Number of pages	Total lines	Detected lines	Text type	Accuracy (%)
Younes et al. [30]	90	1000	940	Handwritten	94
Din et al. [31]	30	310	306	Printed	98.7
Ahmad et al. [21]	47	607	602	Printed	99.17
Proposed method	80	687	674	Handwritten	96.7
Proposed method	48	495	491	Printed	98.3

sequential process (left to right), in which the algorithm first removes the skewness of an image (Figure 7(b)) and then segments it into lines. After segmentation of page into lines, the projection profile method is used for text line segmentation and lines are transformed into words through the vertical profile method. The boundary of each connected dark region in the profile is extracted as a separation region. Ligatures/subwords are segmented as an output of the algorithm, which is shown in Figure 8(b).

4.2. Comparison with Previous Work. Despite prevalent contributions in Urdu OCR, there is no such versatile dataset available for text line segmentation which covers the handwritten and printed text.

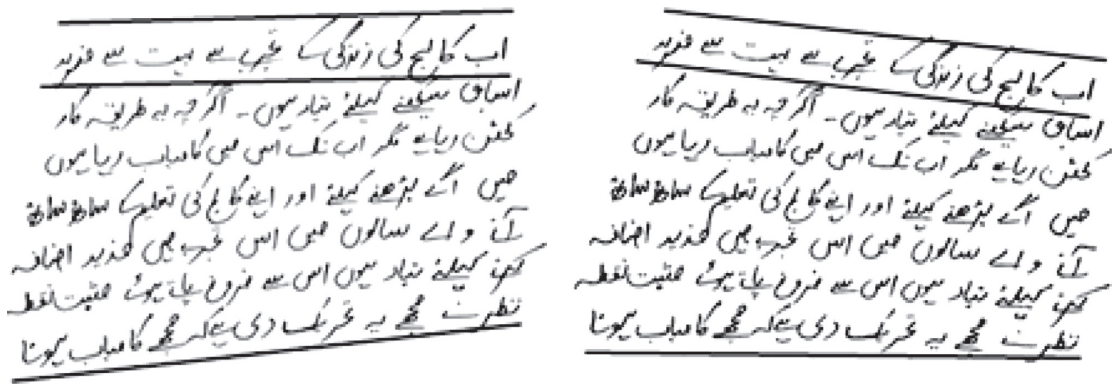
Table 6 shows the previous relevant works with their accuracies. Most works in the OCR field have not used any available dataset; as mentioned in [21, 32, 33], they used their datasets. Therefore, the accuracy of the algorithm depends on their dataset. Mainly in Urdu text, dot/diacritics allocation and skew detection are two issues. The algorithm presented in [21] handles the dot/diacritics allocation problem but does not work well for skew documents. The proposed algorithm overcomes this issue, by using Algorithm 2.

4.3. Results Discussion. Skew detection algorithm easily corrects the skewness of the image if the angle between lines is the same or has a little variation. The result of segmentation suffers when the angle between lines is changing throughout the image as shown in Figure 9(a). As the algorithm is not rotating each line separately, the whole image is rotated as shown in Figure 9(b).

Mainly, two types of issues have occurred during line segmentation, which are over-segmentation and missed/under-segmentation. Partially segmented lines are considered as false detection. Accuracy of segmentation is calculated as either "detected lines" or "not detected lines," because when one line is wrongly segmented into two lines (Figure 10), then it is not recognized in the latter stages.

Inter-line skewness causes under-segmentation. If the paragraph has multi-skewed lines, the algorithm bypasses these lines because the algorithm is unable to detect multi-skewed lines and segments both lines as one line. This issue causes false segmentation of lines as shown in Figure 11.

Over-segmentation is mainly due to dot and diacritics presence. In the proposed algorithm, when the number of dark pixels exceeds the threshold limit, then it is considered as a text line. As given in Figure 10, one line is over-segmented; these lines are false detected lines and considered as incorrect



(a) (b)

Figure 9: (a) Original multi-skewed text image, (b) image after skew correction.

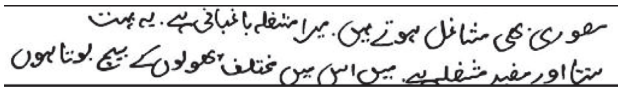


Figure 10: over-segmentation.

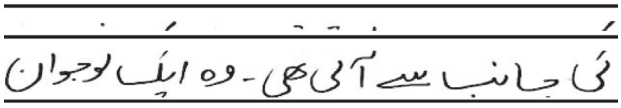


Figure 11: Under-segmentation.

segmented lines. In this way, one line is segmented into two or more lines. This wrong segmentation of the line is known as over-segmentation.

In this research, we focus on handling handwritten text (with and without skewness). This technique has a limitation in segmenting process over dot and diacritics properly.

5. Conclusion

Recently, many script-dependent algorithms for line and ligature have been proposed. But in this research, we put efforts to step forward to propose an efficient algorithm that deals with both printed and handwritten text. The proposed algorithm works well on both printed and handwritten Urdu documents. In the handwritten text, lines are not straight and have a variable size. Especially for handwritten text, the algorithm deals with skewed pages and variable text line size. In the proposed method, the page is preprocessed using an adaptive thresholding technique. Then, the image is rotated if it is skewed and the lines of the de-skewed image are segmented. The proposed line segmentation algorithm is based on counting pixel density (black and white) in a row. The header line and baseline of the text line are determined and segmented. After line segmentation, projection profile technique is used to segment ligatures from the segmented line. The proposed line segmentation algorithm shows promising results on handwritten and printed Urdu text. We make this approach more flexible for handwritten text so that dot/diacritics will remain in the concerned line and not be part of adjacent lines. In this paper, we mainly deal with Urdu text. In the future, we will expand this work as a general technique so that this approach will be applicable to all OCR systems.

Data Availability

All the data used to support the findings of this study are included within the manuscript.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

The authors would like to acknowledge the support by King Saud University, Saudi Arabia, through Researchers Supporting Project number RSP-2020/184.

References

[1] S. A. Malik, M. Muazzam, A. Farhan et al., "An efficient segmentation technique for Urdu optical character recognizer (OCR)," in *Future of Information and Communication Conference*Springer, Berlin, Germany, 2019.

[2] A. F. Ganai and F. R. Lone, "Character segmentation for Nastaleeq Urdu OCR: a review," in *Proceedings of the International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)*, IEEE, Chennai, TN, India, March 2016.

[3] K. Keisham and S. Dixit, "Recognition of handwritten English text U minimisation," in *Information Systems Design and Intelligent Applications*, pp. 607–614, Springer, Berlin, Germany, 2016.

[4] M. Arivazhagan, H. Srinivasan, and S. Srihari, "A Statistical approach to handwritten line segmentation. document recognition and retrieval XIV," in *Proceedings of the SPIE*, pp. 6500T-6501T, San Jose, CA, USA, January 2007.

[5] I. Bar-Yosef et al., "Line segmentation for degraded handwritten historical documents," in *Proceedings of the 10th International Conference on Document Analysis and Recognition ICDAR'09*, IEEE, Barcelona, Spain, July 2009.

[6] C. I. Patel, R. Patel, and P. Patel, "Handwritten character recognition using neural network," *International Journal of Scientific & Engineering Research*, vol. 2, no. 5, pp. 1–6, 2011.

[7] Z. A. Shah, "Ligature based optical character recognition of Urdu-Nastaleeq font," in *Proceedings of the International Multi Topic Conference Abstracts. INMIC 2002*, IEEE, Karachi, Pakistan, February 2002.

[8] M. Ayesh, K. Mohammad, A. Qaroush, S. Agaian, and M. Washha, "A robust line segmentation algorithm for Arabic printed text with diacritics," *Electronic Imaging*, vol. 2017, no. 13, pp. 42–47, 2017.

[9] M. A. Mousa, M. S. Sayed, and M. I. Abdalla, "Arabic character segmentation using projection based approach with profile's amplitude filter," 2017, http://arxiv.org/abs/1707.00800.

[10] G. S. Lehal, "Ligature segmentation for Urdu OCR," in *Proceedings of the 12th International Conference on.Document Analysis and Recognition (ICDAR), 2013*, IEEE, Washington, DC, USA, August 2013.

[11] B. Moysset, K. Christopher, W. Christian et al., "Paragraph text segmentation into lines with recurrent neural networks," in *Proceedings of the 13th International Conference on Document Analysis and Recognition (ICDAR), 2015*, IEEE, Tunis, Tunisia, August 2015.

[12] G. Peng, P. Yu, H. Li et al., "Text line segmentation using Viterbi algorithm for the palm leaf manuscripts of Dai," in *Proceedings of the 2016 International Conference on Audio, Language and Image Processing (ICALIP)*, IEEE, Shanghai, China, February 2016.

[13] Q. N. Vo and G. Lee, "Dense prediction for text line segmentation in handwritten document images," in *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, IEEE, Phoenix, AZ, USA, August 2016.

[14] D. Brodić, "Text line segmentation with water flow algorithm based on power function," *Journal of Electrical Engineering*, vol. 66, no. 3, pp. 132–141, 2015.

[15] S. Marinai and P. Nesi, "Projection based segmentation of musical sheets," in *Proceedings of the Fifth International Conference on Document Analysis and Recognition ICDAR'99*, IEEE, Bangalore, India, October 1999.

[16] D. Brodić and Z. Milivojević, "A new approach to water flow algorithm for text line segmentation," *Journal of Universal Computer Science*, vol. 17, no. 1, pp. 30–47, 2011.

[17] R. Student, "Off-line handwritten Kannada text recognition using support vector machine using zernike moments," *IJCSNS*, vol. 11, no. 7, p. 128, 2011.

[18] A. Choudhary and V. Kumar, "A robust technique for handwritten words segmentation into individual characters," in *Speech and Language Processing for Human-Machine Communications*, pp. 99–106, Springer, Berlin, Germany, 2018.

[19] P. Dhande and R. Kharat, "Segmentation and feature extraction for cursive English handwriting recognition," *IJETT*, vol. 1, no. 2, 2017.

[20] S. Naz, R. Imran, S. Imran et al., "An Ocr system for printed Nasta'liq script: a segmentation based approach," in *Proceedings of the IEEE 17th International Multi-Topic Conference (INMIC)*, IEEE, Karachi, Pakistan, December 2014.

[21] I. Ahmad, X. Wang, R. Li, M. Ahmed, and R. Ullah, "Line and ligature segmentation of Urdu Nastaleeq text," *IEEE Access*, vol. 5, pp. 10924–10940, 2017.

[22] A. Rehman, "An ensemble of neural networks for nonlinear segmentation of overlapped cursive script," *International Journal of Computational Vision and Robotics*, vol. 10, no. 4, pp. 275–288, 2020.

[23] A. Rehman, "Cursive overlapped character segmentation: an enhanced approach," 2019, http://arxiv.org/abs/1904.00792.

[24] A. Qaroush, A. Abdalkarim, M. Mohammad, and Z. Malik, "Segmentation-based, omnifont printed Arabic character recognition without font identification," *Journal of King Saud University-Computer and Information Sciences*, 2020.

[25] K. Mullick, S. Banerjee, and U. Bhattacharya, "An efficient line segmentation approach for handwritten Bangla document image," in *Proceedings of the 2015 Eighth International Conference on Advances in Pattern Recognition (ICAPR)*, IEEE, Kolkata, India, March 2015.

[26] D. Chakraborty and U. Pal, "Baseline detection of multilingual unconstrained handwritten text lines," *Pattern Recognition Letters*, vol. 74, pp. 74–81, 2016.

[27] O. Surinta, L. Schomaker, M. Wiering et al., "A path planning for line segmentation of handwritten documents," in *Proceedings of the 2014 14th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, IEEE, Heraklion, Greece, December 2014.

[28] S. Palakollu, R. Dhir, and R. Rani, "A new technique for line segmentation of handwritten Hindi text," *Special Issue of International Journal of Computer Applications*, vol. 5, pp. 0975–8887, 2011.

[29] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9, no. 1, pp. 62–66, 1979.

[30] M. Younes and Y. Abdellah, "Segmentation of Arabic handwritten text to lines," *Procedia Computer Science*, vol. 73, pp. 115–121, 2015.

[31] I. U. Din, Z. Malik, I. Siddiqi, and S. Khalid, "Line and ligature segmentation in printed Urdu document images," *Journal of Applied Environmental and Biological Sciences*, vol. 6, no. 3, pp. 114–120, 2016.

[32] F. Shafait, D. Keysers, and T. M. Breuel, "Layout analysis of Urdu document images," in *Proceedings of the Multitopic Conference INMIC'06*, IEEE, Islamabad, Pakistan, December 2006.

[33] S. S. Bukhari, F. Shafait, and T. M. Breuel, "High performance layout analysis of Arabic and Urdu document images," in *Proceedings of the 2011 International Conference on Document Analysis and Recognition (ICDAR)*, IEEE, Beijing, China, September 2011.