

Received April 5, 2017, accepted April 30, 2017, date of publication May 19, 2017, date of current version July 3, 2017.

Digital Object Identifier 10.1109/ACCESS.2017.2703155

# Line and Ligature Segmentation of Urdu Nastaleeq Text

IBRAR AHMAD<sup>1</sup>, XIAOJIE WANG<sup>1</sup>, RUIFAN LI<sup>1</sup>, MANZOOR AHMED<sup>2</sup>, AND RAHAT ULLAH<sup>3</sup>

<sup>1</sup>Center for Intelligence of Science and Technology, Beijing University of Posts and Telecommunications, 100876 Beijing, China

<sup>2</sup>Department of Electronic Engineering, FIB Lab, Tsinghua University, 100084 Beijing, China

<sup>3</sup>Nanjing University of Information Science and Technology, 210044 Nanjing, China

Corresponding author: Ibrar Ahmad (toibrar@yahoo.com)

This work was supported in part by the National Natural Science Foundation of China under Project 61273365 and in part by 111 Project under Grant B08004.

**ABSTRACT** The recognition accuracy of ligature-based Urdu language optical character recognition (OCR) systems highly depends on the accuracy of segmentation that converts Urdu text into lines and ligatures. In general, lines and ligatures-based Urdu language OCRs are more successful as compared to characters-based. This paper presents the techniques for segmenting Urdu Nastaleeq text images into lines and subsequently to ligatures. Classical horizontal projection-based segmentation method is augmented with a curved-line-split algorithm for successfully overcoming the problems, such as text line split position, overlapping, merged ligatures, and ligatures crossing line split positions. Ligature segmentation algorithm extracts connected components from text lines, categorizes them into primary and secondary classes, and allocates secondary components to the primary class by examining width, height, coordinates, overlapping, centroids, and baseline information. The proposed line segmentation algorithm is tested on 47 pages with 99.17% accuracy. The proposed ligature segmentation algorithm is mainly tested on a large Urdu-printed text images data set. The proposed algorithm segmented Urdu-printed text images data set to 189 000 ligatures from 10 063 text lines having 332 000 connected components. A total of about 142 000 secondary components have been successfully allocated to more than 189 000 primary ligatures with accuracy rate of 99.80%. Thus, both of the proposed segmentation algorithms outperform the existing algorithms employed for Urdu Nastaleeq text segmentation. Moreover, the proposed line segmentation algorithm is also tested on Arabic, for which it also extracted lines correctly.

**INDEX TERMS** Urdu text segmentation, Nastaleeq script segmentation, line and ligature segmentation, preprocessing.

## I. INTRODUCTION

Urdu is the national language of Pakistan, and is widely spoken and understood in India and other Southern Asian countries as well. It is one of the top ten influential languages (weber)<sup>1</sup> of world. The number of research work is ongoing to preserve its literature. Urdu is an Arabic script language, mostly written in Nastaleeq writing style. Nastaleeq writing style is also followed in many other languages including Persian, Pashto, Panjabi, Baluchi, and Siraiki [1], [2]. Successful Optical Character Recognition (OCR) for Nastaleeq text therefore becomes imperative.

Generally, Urdu language OCR systems can be divided into three categories based upon text's segmentation, which

includes character based [4], [10]–[17], ligature based [3], [18], [30], [31] and sentence based systems [20]–[23]. Currently, researchers are focusing more on ligature and sentence based OCRs. All these OCR systems mainly rely on of document images' segmentation into text lines, ligatures and characters. Complexities of Urdu Nastaleeq writing style are the major obstacles for its segmentation into lines and ligatures. Major complexities of Nastaleeq writing style include context sensitiveness, compactness, overlapping, cursive nature, diagonality, and many others [1] as presented in Fig. 1.

Nastaleeq accommodates more characters in less space because of its diagonality and cursive nature [1]. Characters are stacked from right to left and top to bottom into a word.

For instance, the starting character of each ligature in Fig. 2 is same but different in shape [4]. In case of Urdu Nastaleeq

<sup>1</sup><http://www2.ignatius.edu/faculty/turner/languages.htm> (accessed 10/05/2016, 4:25am)

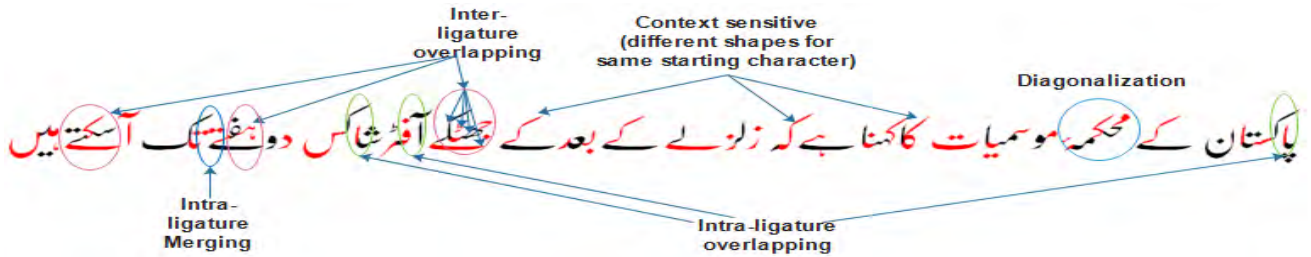


FIGURE 1. Complexities of Nastaleeq writing style.



FIGURE 2. Different shapes of same starting character of words in Nastaleeq.

writing style, the number of possible shapes of certain characters reach up to 60 [5], [6]. Therefore, it is very difficult to segment text in Nastaleeq writing style due to large number of character shapes, inter-ligature and intra-ligature overlaps, and context sensitivity of text [2].

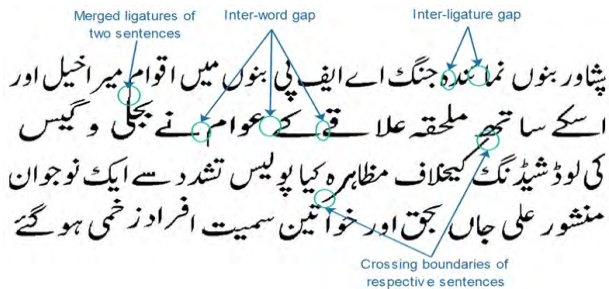


FIGURE 3. Paragraph of Nastaleeq writing style and its complexities.

Furthermore, in Nastaleeq writing style, there is no consistency in spacing among the words, ligatures and sentences. Some ligatures of sentences intrude the boundaries of other sentences that are written above or below lines, as presented in Fig. 3.

In the first line of the paragraph, the character “نہ” is connected with the word “نہ” of second line. Besides, ligature “نہ” of third line is entering in the boundary of second line. Also, ligature “نہ” of third line is crossing the border of fourth line. Thus, in Urdu Nastaleeq text, there exists no fixed inter ligature, inter words and inter sentences gap as demonstrated in Fig. 3.

#### A. PREVIOUS WORK

An overview of contributions in the area of Urdu line and ligature segmentation is briefed in the following subsections.

#### 1) LINE SEGMENTATION

Text lines extraction is a very crucial step in segmenting procedure. The algorithms applied for Urdu page segmentation into lines include horizontal and vertical projection, x-y cut, smearing, geometric, ridge based, and zone-based horizontal projection algorithms.

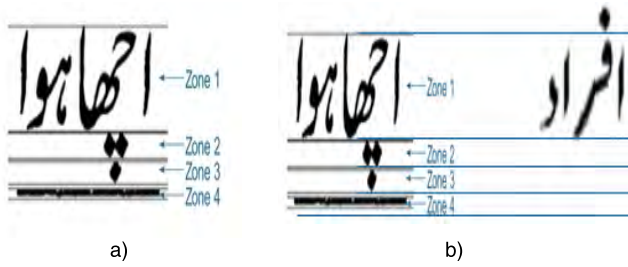
The pioneer work [4] on Urdu OCR used horizontal projection profile method for text lines extraction, followed by component labeling and vertical projection profile methods for character extraction. Horizontal and vertical projection methods were applied on scanned document image for text lines and character segmentation, respectively [7], [25]. Although, horizontal projection segmented the text lines correctly but it could not segment lines with merged ligatures.

Recursive XY cut, whitespace analysis, constrained text-line detection, docstrum, voronoi-diagram based algorithm, and smearing algorithm were used [26] on five scripts including Nastaleeq were segmented into text lines, and the highest accuracy reported for Urdu was 65.4%. The results are not promising for Urdu text, because of its overlapping nature, non-equispaced lines and non-availability of clear demarcation.

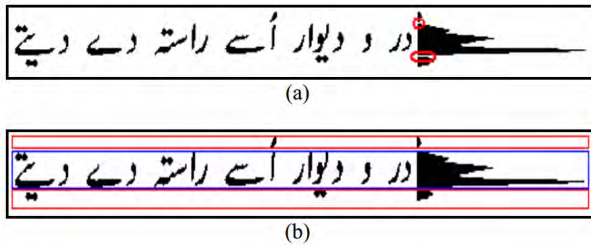
Geometric algorithms for layout [27] were adopted and tailored for getting the correct layout of Urdu Nastaleeq documents [28]. The authors achieved accuracy of text line detection ranging from 72% to 93% for different genres of document images. Ridge based approach [29], [33]–[35] was applied for text line extraction, while the method in [28] was adopted for reading order. The achieved accuracies for Arabic and Urdu document images were 96% and 92%, respectively. Although these methods are accurate but lack proper allocation of misplaced diacritics/dots.

Zone based global horizontal projection method [8] was proposed for segmenting Nastaleeq's Urdu text lines. The method segmented document images containing touching ligatures across lines with the accuracy of 99.11%. In this method, there was no zone mentioned above the zone 1, as depicted in Fig. 4(a). Therefore, it overlooks the diacritics of the text line that appear above zone 1, as presented in Fig. 4(b).

In Urdu Nastaleeq, it is possible to have some diacritics/dots above the zone 1 (i.e.; main text). So, there are possibilities of assigning diacritics/dots inaccurately, as depicted in Fig. 4(b). Implicitly, three zones are suggested [7]



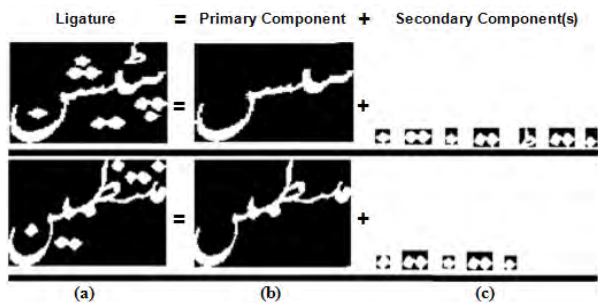
**FIGURE 4.** Zone-based Urdu Nastaleeq text line segmentation. a) Zone based line segmentation[8]. b) Diacritics/dots above zone 1.



**FIGURE 5.** Zone above zone 1 [7].

for handling 4 zones for page segmentation as shown in Fig. 5.

This strategy takes into consideration the dots/diacritics even above zone 1. A recent study [9] also used zone based method [8] with dilating document images before segmentation for getting more correct size of zones, which resulted in 98.7% line segmentation accuracy. This method confronts difficulties for segmenting merged text lines.



**FIGURE 6.** Primary and secondary components of two ligatures.

## 2) LIGATURE SEGMENTATION

A ligature is composed of one or more characters joined together [8], and the combination of one or more ligatures form words in Arabic script (like Urdu language). A ligature is comprised of two parts; the primary component is made up of main stroke, while diacritics are nominated as secondary strokes as presented in Fig. 6.

Considerable efforts were devoted for line segmentation into ligatures like vertical histogram method [20] and [27], horizontal projection method [4], and bounding box method [8], [9], [18], [28], [31].

Individual ligatures extracted by vertical histogram of text line in Naskh was described in [20] and [27], but this method is not applicable for Nastaleeq because of inter-ligature overlapping. In another work [4], vertical projection was applied for getting ligatures and character on extracted lines by horizontal projection method. Ligatures were also segmented by connected component method supplemented with baseline and centroid information, while achieving 94% segmentation accuracy [31]. Connected components were extracted by bounding box methods from text lines [18] and [28], and then secondary components were allocated to primary components based on the detected baseline. Text lines were segmented into connected components that were categorized into 6 types [8], and then secondary components were merged together with primary components to get the resultant ligatures based upon certain heuristics, thus, achieving 99.02% accuracy. Another study [9] first segmented lines into connected components, and then allocated secondary components to primary components with 92.5% accuracy.

## B. MOTIVATION

The most recent line and ligature segmenting algorithms [8], [9] proposed the use of 4 zones for line segmentation, and 6 categories of connected components for constructing a final ligature. The authors did not utilize the baseline information that is proved to be very much decisive in our proposed line segmentation algorithm. Also, ligature segmentation [8] relies on 6 heuristics, among which only one heuristic uses baseline information. Relying more on zonal and heuristics information for line and ligature segmentation becomes very much context and font dependent. Lines were segmented with perfection [28], [29], but misplaced dots/diacritics were not allocated to their respective lines. Thus, our motivation is to reduce the complex mechanism of zoning and their merging decisions for line segmentation. In our approach, we mainly rely on baseline information with three zonal method for dots/diacritics allocation, and quantitative values (width, height, centroids, and overlapping) for ligature segmentation.

## C. CONTRIBUTION

In this paper for Urdu Nastaleeq text, we present two algorithms for line and ligature segmentation. These algorithms can also be applied to other Nastaleeq based scripted languages, such as Arabic, Persian, Pashto, and Sindhi etc. Specifically, our proposed line segmentation algorithm depends solely on horizontal projection, split positions (proposed in this paper and named as Curved-Line-Split (CLS) algorithm) and baseline information. In proposed approach, the allocation of dots/diacritics to proper text line is also properly addressed by employing three-zonal method. For ligature segmentation, the proposed algorithm employs width, height, centroids, and overlapping quantitative information as opposed to heuristics [8], [9]. The accuracies achieved by both the proposed line and ligature segmentation algorithms are 99.17% and 99.08%, respectively, which are better than the previous contributions [8], [9], [29].

**Algorithm 1** Curved Line Split**Data:** Document Image Page, initial split row numbers**Result:** Coordinates for demarcation/split lines**begin**

Draw baselines for every text line. Baselines will be the maximum intensity *pixels'* row between every two consecutive initial split row numbers.

**for** *each initial split row number* **do**

$x$  = initial split row number,  $y=1$ , set *resolve*=0, *stuck*=0.

Store ( $x,y$ ) as starting coordinates for demarcation line.

$x$  = initial split  $n^{th}$  row number,  $y=1$ .

Direction to move up or down when find an obstacle= $d$ =down.

$bl1$ = baseline of current line,  $bl2$  = baseline of previous line.

**while**  $y \leq \text{Pages Max column}$  and  $bl1 < x < bl2$  **do**

Move forward on  $x^{th}$  row by increasing  $y$  until find an obstacle(text pixels).

Store ( $x, y - 1$ ) as coordinates for demarcation line.

Move in  $d$  direction through column ( $y - 1$ ) until find an obstacle (text pixels).

$d$ =opposite of previous direction.

Store ( $x/2, y - 1$ ) as coordinates for demarcation line.

**if**  $pre_y = y - 1$  **then**

└  $stuck = stuck + 1$

$pre_y = y - 1$

**if** *stuck*(No change in value of  $y$ ) for 4 times and *resolve* < 4 **then**

└ go back 3 pair values in coordinates for demarcation line

**if** *current rows* > *next rows*  $x$  in coordinates for demarcation line **then**

└ set  $d$ =down

**else**

└ set  $d$ =up

set  $x = x$  of previous row coordinates for demarcation line.

set  $y = y$  of previous col coordinates for demarcation line.

$pre_y = y$

set *resolve* = *resolve* + 1 and *stuck* = 0.

**if** *stuck*(no change in value of  $y$ )=4 and *resolve*=4 **then**

└ Move through info pixels on  $x$  row and increase  $y$  until information pixels are over.

└ Set *resolve* = 0 and *stuck* = 0,  $pre_y = y - 1$ .

└  $y = y + 1$ .

Return coordinates for demarcation/split lines

The remaining paper is organized as follows. In Section II, line segmentation algorithm is discussed. While, in Section III we discuss the ligature extraction algorithm. In Section IV, results are analyzed for evaluating the performance of the proposed algorithms. Finally, Section V includes the concluding remarks.

**II. LINE SEGMENTATION**

This section details about segmentation algorithm of a page into constituents text lines, which employs global horizontal projection method augmented with the proposed CLS algorithm. Additionally, it contains steps for conflicts resolution regarding components crossing split lines based upon baseline information.

**A. CURVED LINE SPLIT ALGORITHM**

The proposed CLS algorithm can even find out a curved (not straight) demarcation line between every two

consecutive text lines, when there is no straight demarcation line. The basic idea of CLS algorithm is as follows:

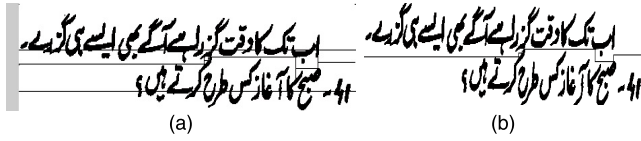
**Basic Idea:** Algorithm starts a demarcation line from middle of baselines of consecutive text lines. It traverses a path from left to right on horizontal path, if it finds text pixels it moves up or down vertically until it finds a text pixel or baseline, and then from half of the vertical movement it starts again horizontal traversal in between two lines.

The procedure of CLS is presented in Figs. 7 and 8. The CLS algorithm detail is presented in Algorithm 1.

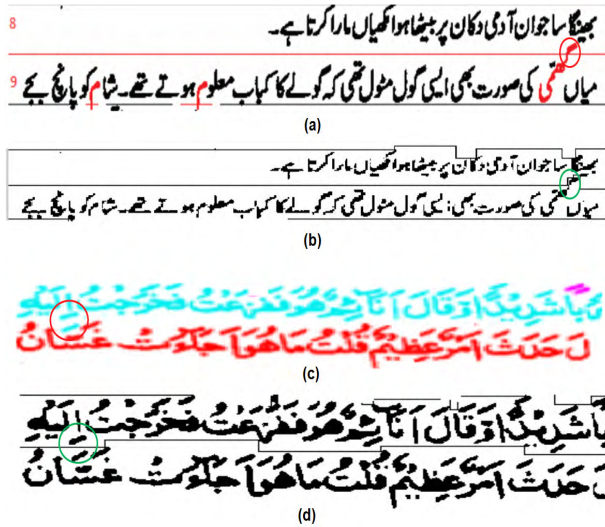
**B. LINE SEGMENTATION ALGORITHM**

Line segmentation algorithm is applied on Urdu Nastaleeq document image for getting segmented text lines. Classical horizontal projection based segmentation method is augmented with a proposed CLS algorithm. This augmentation successfully overcomes the problems, such as text line split





**FIGURE 7.** Application of CLS algorithm and its output with demarcation line. (a) Working of CLS algorithm. (b) Demarcation line, output of CLS algorithm.



**FIGURE 8.** Improper segmentations encircled red in (a)[8], (c)[29] are solved by the proposed Curved Line Split algorithm encircled green in (b) and (d).

position, overlapping, merged ligatures and ligatures crossing line split positions. Horizontal line segmentation algorithms [4], [7]–[9] can split two text lines if there is a clear horizontal demarcation. The CLS algorithm overcomes this flaw and can find a demarcation boundary in case of non-equispaced text lines and non-availability of clear demarcation. The main steps are described as follows:

1. Find Line-Peaks: In this step, an image of Urdu Nastaleeq script  $I_U$  is converted into binary image  $I_{UB}$  by global thresholding method. This binarized image is inverted for getting an image  $I'_{H \times W}$  with white text on black background. Then, horizontal projection (HP) is calculated for the entire document.

$$HP(x) = \sum_{y=1}^W I'(x, y), \quad \forall x \in [1, H] \quad (1)$$

Find all peaks and troughs of horizontal projection.

$$\text{Peaks} : \forall x \text{ find } x_i \text{ s.t. } x_{i-1} < x_i > x_{i+1} \quad (2)$$

$$\text{Troughs} : \forall x \text{ find } x'_i \text{ s.t. } x'_{i-1} > x'_i < x'_{i+1} \quad (3)$$

First of all peaks are scanned to find out all local highest valued peaks. Generally, such peaks start with a lowest peak value after a trough followed by step-wise higher valued peaks until we get a local highest valued peak. Then, the values of next coming peaks gradually decrease until it reaches another trough. Such local

highest valued peaks are termed as Line-Peaks, which are accompanied by gradual lower valued peaks on both sides representing one line. Line-Peaks represent center/base with maximum pixel intensity of text lines in a page image.

Line – Peaks :  $\forall x \text{ find all } x''_c \text{ s.t.}$

$$x_{i-1} < x''_c > x_{i+1} \quad (4)$$

2. Next, we find the average number of pixels' rows per text line.

$$\text{Average Text Line (TL) Height: } Avg_H(TL) = \frac{|x|}{|x''|},$$

Where,  $Avg_H(TL)$  is Average Text Line (TL) Height,

$|x|$  represents total rows of pixels,

$x''$  represents number of Line\_Peaks, (5)

3. False text lines: Once again scan the peaks, if lowest troughs of some Line-Peak come before  $(3/4)^{\text{th}}$  of the average text line height, then, that line is considered as false Line-Peak and excluded from total Line-Peaks.

$$\forall x'' \text{ find} : a = |x''_{ci-1} - x''_{ci-2}|,$$

$$b = |x''_{ci} - x''_{ci-1}|,$$

$$c = |x''_{ci+1} - x''_{ci}|,$$

$$\text{if } c < \frac{3}{4} Avg_H(TL) \wedge a < c, \quad a = a + b,$$

$$\text{Eliminate } x''_{ci} \text{ from Line – Peaks} \quad (6)$$

4. Missed text lines: Fig. 12 depicts that all Line-Peak values must be greater than the highest trough value; otherwise the text lines with Line-Peaks less than the highest trough will be missed. After conforming step 3, we hypothetically increase the value of all such Line-Peaks more than the highest trough value. In this way, we fix the problem of missed lines.

Highest Trough :  $\max x'_T = H_T \forall x \text{ find } x'_i \text{ s.t.}$

$$x'_{i-1} > x'_i < x'_{i+1}] \quad (7)$$

Missed Text Lines :  $\forall x''_{ci} < H_T,$

$$x''_{ci} = x''_{ci} + (|H_T - x''_{ci}|) \quad (8)$$

5. Segment page image into text lines: Pass initial split row numbers to CLS algorithm. The initial candidate split row numbers are the lowest trough values between every two Line-Peaks. Convert the text page into connected components format, so that every component is represented by a different number. Then, split underlying page according to the coordinates for demarcation of lines returned by CLS algorithm.

6. Assignment of components: Assignment of misplaced and crossing components to proper text line can easily be done with the information of baseline of every text line. Components that are crossing the demarcation line are referred as crossing components. For visualization of this concept, four cases are briefed as follows:

Case 1: Crossing component touching one baseline: Move the crossing component to the text line where it is touching the baseline.

Case 2: Crossing component not touching any baseline:

- a) Here, find the top row, bottom row, left most column and right most column numbers for under consideration crossing component.
- b) Next, start search by moving above from the top row and below from the bottom row while checking every pixel from left most column to right most column for finding the nearest text pixel of any other component.
- c) Move this crossing component to the corresponding side where text pixel is found earlier. In case of tie, move component to the side where search found component is touching baseline. If both are not touching the baseline, then perform step a, b and c for both newly found components, but search is performed in the same direction where these new components are found.

Case 3: Connected component touching neither base lines nor split line: Some diacritics/dots might neither crossing any base nor split line. Such components might be segmented in wrong text line. Therefore, scan every text line for any text component measuring approximately  $(1/4)^{\text{th}}$  height of each line. These areas are referred as upper and lower diacritics/dots zones as presented in Fig. 9.

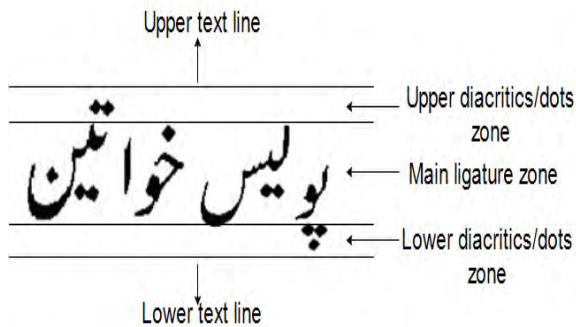


FIGURE 9. Upper and lower diacritics/dots zones scanning.

There may be possibilities of attaching diacritics to wrong lines, most probably in upper and lower diacritics/dots zones of every text line except first and last ones. First text line may only have lower diacritics/dots zone, while last one may have only upper diacritics/dots zone.

In these portions, if some component is found then it is moved to the line where it has minimum distance from the neighboring component. This step can easily be performed as follows:

- a) For every component found in upper or lower diacritics/dots zone, where:
  - i. The component must not touch baseline.
  - ii. The component must not be greater in size than its respective zone.

- b) Find the top and bottom row of each component. Also, find the left and right most column numbers of its maximum span/width.
- c) Start searching above and below from the component's top coordinates and bottom coordinates, respectively, for finding the nearest text pixel.
- d) Move this component to line where text pixel is found earlier. In case of tie, move the component to the line where search finds the component that is touching baseline. If both are not touching baseline then for breaking the tie perform step a, b and c for both newly found components but search is performed in the same direction where these new components were found.

Case 4: Crossing component touching both baselines: This situation comes along when two ligatures are connected mistakenly. We need to find the position of the joint of two ligatures with minimum number of pixel. Start from the split position of text lines along the crossing component above and below, and search for minimum pixel connectivity nearby split position. This search must not move more than one fourth of the height of a text line. Otherwise, split this component from the text line split position.

Examples: Each step of the proposed line segmentation algorithm is explained by applying it on a complex text page presented in [28], as shown in Fig. 10(a). Also, algorithm is applied on other sample pages in earlier works [8], [9], [29] are shown in Fig 10 (b), (c) and (d), respectively.

Steps 1-3: Top most Line-Peak/baseline in Fig. 11(a) represents false Line-Peak that may result a false text line. As, Fig. 11(b) represents Line-Peak and its corresponding troughs last well before  $(3/4)^{\text{th}}$  of average text line height, so it is eliminated from Line-Peaks. Furthermore, this eliminated Line-Peak is merged with the smaller height neighbored line.

Step 4: Fig. 12 depicts that all Line-Peaks less than the highest trough will be missed, as last line of the page is a missed line.

Step 5: Initial lines split after demarcation by CLS algorithm, as shown in Fig. 13.

Step 6: Resolution of 4 cases is shown in Fig.14-16:

The proposed line segmentation algorithm extracted all constituent text lines with only two diacritics out of place encircled red as indicated in Fig. 17.

### III. LIGATURE SEGMENTATION

Ligature segmentation algorithm splits text lines into constituent ligatures. A ligature is usually composed of 1 to 8 characters forming one primary and zero or more secondary connected components. Main body of a ligature is known as primary component that is made up of longest primary stroke. Secondary components are usually dots and diacritics, as presented in Fig. 6. The basic idea of our algorithm is to find all components and then allocate secondary components to proper primary component as illustrated in Algorithm 2.



This step classifies the connected components into primary and secondary on the basis of collected information in previous step. It is depicted in Algorithm 4.



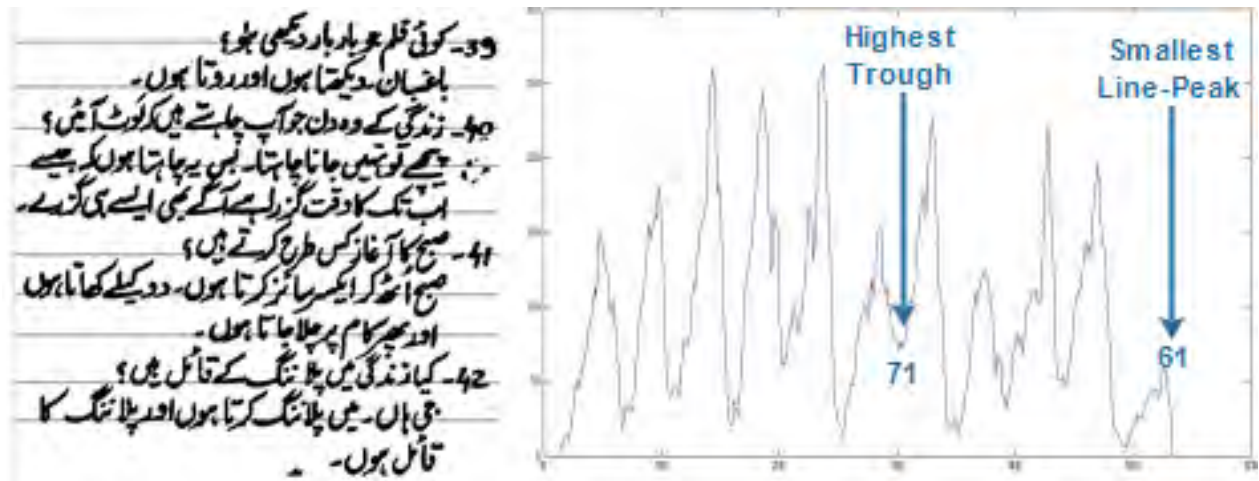


FIGURE 12. Last line represents missed Line-Peak, as its value 61 is less than highest trough value 71.

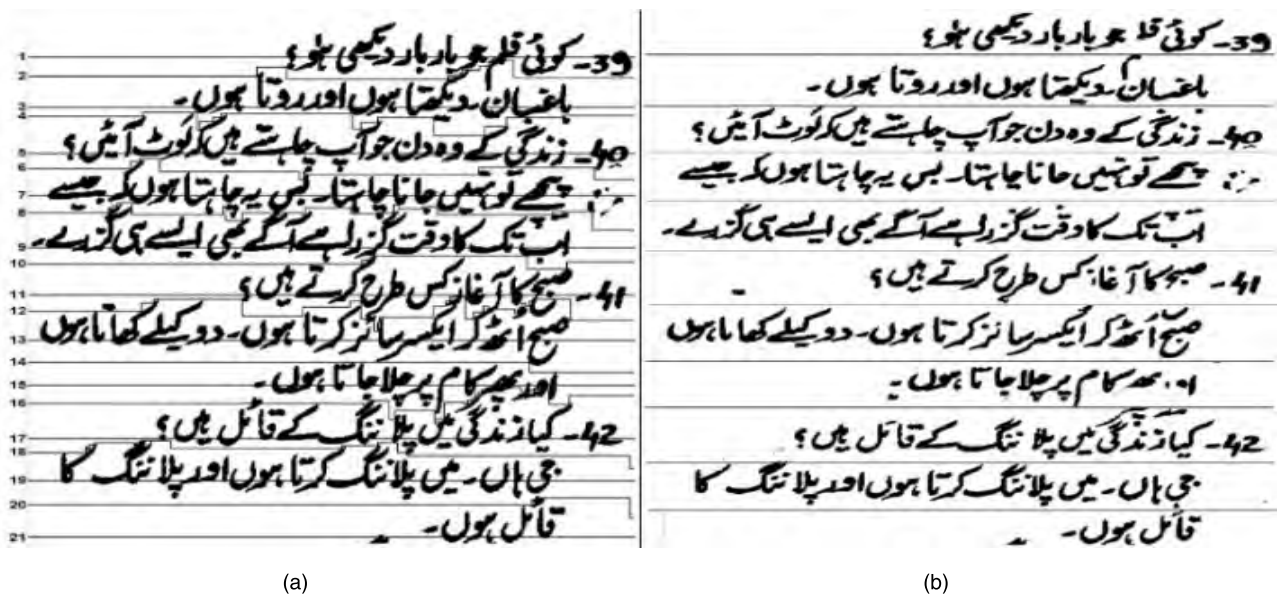


FIGURE 13. Step 5, Fig. 13(a), odd and even numbered lines are representing base and split lines, respectively. Initial segmented text lines are in Fig. 13(b).

### C. ALLOCATE SECONDARY TO PRIMARY COMPONENTS

In Algorithm 5, pixels of secondary components are moved to their respective primary. Firstly, all the non-conflicting (not touching baseline) secondary components are moved to respective primary components. The conflicting secondary components (touching the baseline) are decided based on their size and height. If size/height is less than the threshold size/height of secondary components, then these components are declared as secondary and copied to the respective primary components, otherwise, these components are declared as primary. The execution steps of this mechanism are depicted in Algorithm 5.

The proposed ligature segmentation algorithm is applied to the sentences of UPTI data set as well as to the segmented sentences presented in Section II. For example, three sentences and corresponding ligatures are depicted in Fig. 18.

## IV. RESULTS AND ANALYSIS

In this section, the results of the proposed line and ligature segmentation algorithms are discussed. Results are compared with the prevalent studies.

### A. RESULTS OF THE LINE SEGMENTATION ALGORITHM

The proposed line segmentation algorithm is tested on 47 pages. Lines are segmented 100% for 7 pages taken from earlier papers [8], [9], [28], [29]. Remaining 40 pages are taken from single columned editorials of an online Urdu newspaper.

The proposed line segmentation algorithm segmented lines of 47 pages with the accuracy of 99.17% that is better than accuracies reported in prevalent work. Overall accuracy of the proposed line segmentation algorithm is 99.17% as presented in Table 1.



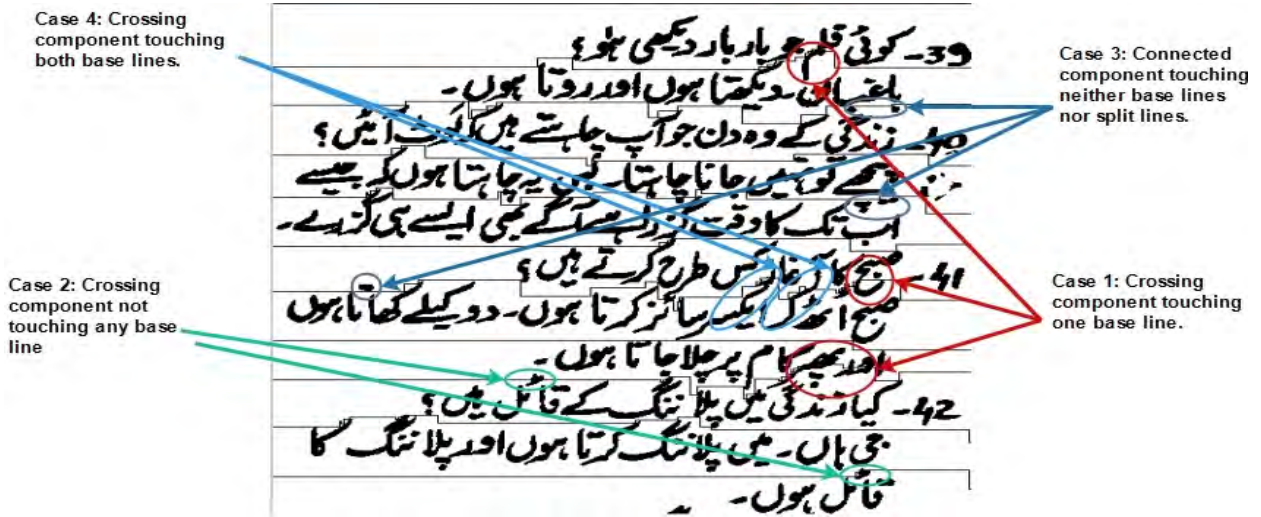


FIGURE 14. Four cases for resolution of conflicting components.

<p>39- کوئی قلم جو بار بار دیکھی ہو؟</p> <p>بعض اوقات دیکھتا ہوں اور دوتا ہوں۔</p> <p>جو زندگی کے وہ دن جو آپ چاہتے ہیں کہ کوٹ آئیں؟</p> <p>میں تو نہیں جانتا چاہتا ہوں۔ یہ چاہتا ہوں کہ جیسے</p> <p>اب تک کا وقت گزرے جس کے لیے ہی گزریں۔</p> <p>41- صبح کا آؤ گس طرح کرتے ہیں؟</p> <p>صبح آؤ گس طرح کرتا ہوں۔ دو کیلے کھانا ہوں</p> <p>اور صبح کا کام پر چلا جاتا ہوں۔</p> <p>42- کیا زندگی میں پلاننگ کے قائل ہیں؟</p> <p>جی ہاں۔ میں پلاننگ کرتا ہوں اور پلاننگ کا قائل ہوں۔</p> <p>(a)</p>	<p>39- کوئی قلم جو بار بار دیکھی ہو؟</p> <p>بعض اوقات دیکھتا ہوں اور دوتا ہوں۔</p> <p>جو زندگی کے وہ دن جو آپ چاہتے ہیں کہ کوٹ آئیں؟</p> <p>میں تو نہیں جانتا چاہتا ہوں۔ یہ چاہتا ہوں کہ جیسے</p> <p>اب تک کا وقت گزرے جس کے لیے ہی گزریں۔</p> <p>41- صبح کا آؤ گس طرح کرتے ہیں؟</p> <p>صبح آؤ گس طرح کرتا ہوں۔ دو کیلے کھانا ہوں</p> <p>اور صبح کا کام پر چلا جاتا ہوں۔</p> <p>42- کیا زندگی میں پلاننگ کے قائل ہیں؟</p> <p>جی ہاں۔ میں پلاننگ کرتا ہوں اور پلاننگ کا قائل ہوں۔</p> <p>(b)</p>	<p>39- کوئی قلم جو بار بار دیکھی ہو؟</p> <p>بعض اوقات دیکھتا ہوں اور دوتا ہوں۔</p> <p>جو زندگی کے وہ دن جو آپ چاہتے ہیں کہ کوٹ آئیں؟</p> <p>میں تو نہیں جانتا چاہتا ہوں۔ یہ چاہتا ہوں کہ جیسے</p> <p>اب تک کا وقت گزرے جس کے لیے ہی گزریں۔</p> <p>41- صبح کا آؤ گس طرح کرتے ہیں؟</p> <p>صبح آؤ گس طرح کرتا ہوں۔ دو کیلے کھانا ہوں</p> <p>اور صبح کا کام پر چلا جاتا ہوں۔</p> <p>42- کیا زندگی میں پلاننگ کے قائل ہیں؟</p> <p>جی ہاں۔ میں پلاننگ کرتا ہوں اور پلاننگ کا قائل ہوں۔</p> <p>(c)</p>	<p>39- کوئی قلم جو بار بار دیکھی ہو؟</p> <p>بعض اوقات دیکھتا ہوں اور دوتا ہوں۔</p> <p>جو زندگی کے وہ دن جو آپ چاہتے ہیں کہ کوٹ آئیں؟</p> <p>میں تو نہیں جانتا چاہتا ہوں۔ یہ چاہتا ہوں کہ جیسے</p> <p>اب تک کا وقت گزرے جس کے لیے ہی گزریں۔</p> <p>41- صبح کا آؤ گس طرح کرتے ہیں؟</p> <p>صبح آؤ گس طرح کرتا ہوں۔ دو کیلے کھانا ہوں</p> <p>اور صبح کا کام پر چلا جاتا ہوں۔</p> <p>42- کیا زندگی میں پلاننگ کے قائل ہیں؟</p> <p>جی ہاں۔ میں پلاننگ کرتا ہوں اور پلاننگ کا قائل ہوں۔</p> <p>(d)</p>	<p>39- کوئی قلم جو بار بار دیکھی ہو؟</p> <p>بعض اوقات دیکھتا ہوں اور دوتا ہوں۔</p> <p>جو زندگی کے وہ دن جو آپ چاہتے ہیں کہ کوٹ آئیں؟</p> <p>میں تو نہیں جانتا چاہتا ہوں۔ یہ چاہتا ہوں کہ جیسے</p> <p>اب تک کا وقت گزرے جس کے لیے ہی گزریں۔</p> <p>41- صبح کا آؤ گس طرح کرتے ہیں؟</p> <p>صبح آؤ گس طرح کرتا ہوں۔ دو کیلے کھانا ہوں</p> <p>اور صبح کا کام پر چلا جاتا ہوں۔</p> <p>42- کیا زندگی میں پلاننگ کے قائل ہیں؟</p> <p>جی ہاں۔ میں پلاننگ کرتا ہوں اور پلاننگ کا قائل ہوں۔</p> <p>(e)</p>
---	---	---	---	---

FIGURE 15. Red encircled components of previous step are resolved in next step, encircled green, (a) to (b) crossing component touching one baseline, (b) to (c) crossing component does not touching any baseline, (c) to (d) component neither touching baseline nor split lines, (d) to (e) component touching 2 baselines.

<p>Original</p> <p>Detached</p> <p>a) Components detached that were touching across line 4 and 5</p>	<p>Original</p> <p>Detached</p> <p>b) Components detached that were touching across line 6 and 7</p>	<p>Original</p> <p>Detached</p> <p>c) Components detached that were touching across line 6 and 7</p>
--	--	--

FIGURE 16. Example of crossing components, touching both baselines.

## B. COMPARISON OF LINE SEGMENTATION ALGORITHMS

Prevalent contributions in the area Urdu line segmentation used their own data sets, as there is no publically available data set. Therefore, accuracy depends upon the complexity of text images used for segmentation. We have observed very complex text images shown by [28] and [29] in their contributions and highest reported accuracy is 92%, as depicted in Table 2. The authors proposed multicolumn Urdu text line

detection mechanism. However, they have not incorporated mechanism for dots/diacritics allocation. We have used 4 text page images from their work and our algorithm performed better with an accuracy of 99.17% because of diacritics allocation mechanism. Lehal [8] and Din *et al.* [9] reported 99.11% and 98.70% accuracies.

Line segmentation [8] draws a straight boundary line between two text lines that may not always be straight. There-

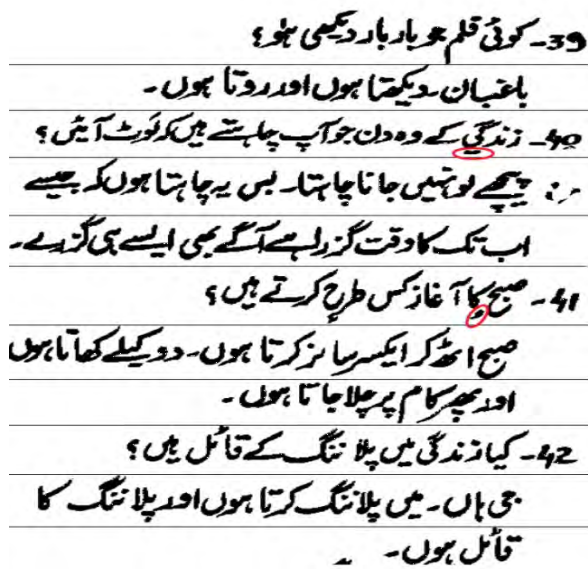


FIGURE 17. Segmented text lines by the proposed line algorithm.

**Algorithm 2** Ligature Segmentation Algorithm**Data:**  $S$  = Urdu Sentence Images**Result:** List of ligature images for a given Urdu sentence image  $S_k$ ,  $L^k = \{Ligature_i^k\}_{i=1}^m$ **begin****for**  $k = 1$  to  $S$  **do** $T^k = \text{Extract information}(\text{connected comp}(S_k))$  $T^k = \text{Decide primary secondary components}(T^k)$  $L^k = \text{Allocate secondary to primary}(T^k)$ **return**  $L^k = \{Ligature_i^k\}_{i=1}^m$ ,  $m = \text{number of ligatures of } S_k$ **Algorithm 3** Extract Information**Data:**  $S_k$  = Urdu sentence image,Connected Component =  $CC$ **Result:** Table  $T^k$ , containing information regarding connected components of a sentence  $S_k$ **begin** $CC = \text{Find connected components for } S_k$ **for each connected components**  $CC_i$  **do**

Find starting and ending columns number

Find starting and ending rows number

Touching baseline or not

Calculate height and width

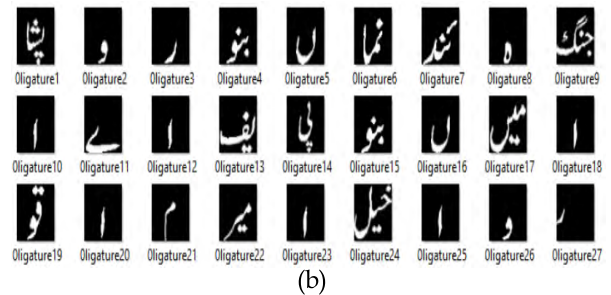
Overlapping connected component

Centroid

Save all above information about  $CC_i$  in  $T_i^k$ **return**  $(T^k)$ 

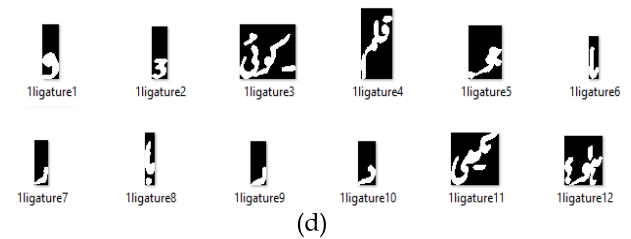
پشاور بنوں نمائندہ جنگ اے ایف پی بنوں میں اقوام میرا خیل اور

(a)



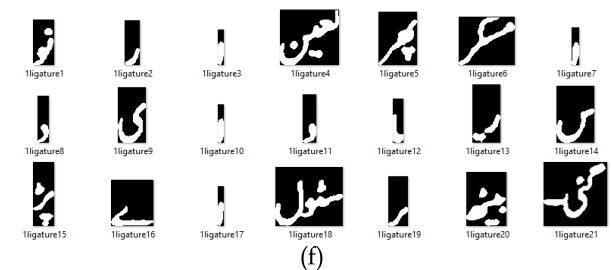
39۔ کوئی قلم جو بار بار دیکھی ہو

(c)



نورالعین پھر مسکرا دی اور پاس پڑے اسٹول رہینہ گئی۔

(e)



**FIGURE 18.** Total 27, 12 and 21 resultant ligatures of above displayed sentences are taken from UPTI data set, [28] and [29], respectively. First sentence of UPTI data set and its segmented ligatures are represented in (a) and (b). A sentence of a text page given in [28] and its segmented ligatures are represented in (c) and (d), respectively. Similarly, a sentence of a text page given in [29] and its segmented ligatures are represented in (e) and (f), respectively.

tial segmentation [8] on a merged text lines page is depicted in Fig. 19(a). CLS algorithm of the proposed line segmentation algorithm is applied on the same page and resultant initial segmented page is depicted in Fig. 19(b).

The line segmentation [8] segments a text page into zones and subsequently zones are segmented into lines by applying straight split line. So, whenever CLS algorithms changes its path for drawing boundary, actually it avoids the later

fore, it needs to place many text components to proper text line later on. The proposed line segmentation algorithm can draw a straight as well as curved boundary. The results of ini-

**TABLE 1.** Results of the proposed line segmentation algorithm.

Source	Script	Pages	Lines	Correctly segmented lines	Accuracy
[8]	Urdu	1	11	11	100%
[9]	Urdu	2	28	28	100%
[28]	Urdu	1	11	11	100%
[29]	Urdu	2	16	16	100%
[29]	Arabic	1	9	9	100%
Newspaper	Urdu	40	532	527	99.06%
Average Accuracy					99.17%

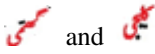
**TABLE 2.** Comparison of line segmentation algorithms.

Source	Pages	Lines	Detected lines	Accuracy
[8]	448	Not reported	Not reported	99.11%
[9]	30	310	306	98.70%
[28]	25	Not reported	Not reported	Books and Magazines= 90% Digest=80% Newspaper= 72%
[29]	20	Not reported	Not reported	92%
This study	47	607	602	99.17%

**TABLE 3.** Initial split row numbers.

Table 3(a) Initial split for Fig. 22(a)			Table 3(b) Initial split for Fig. 22(c)		
Lines #	Intensity value	Row split number	Lines #	Intensity value	Row split number
1&2	42	93	1&2	3	61
2&3	83	164	2&3	19	120
3&4	82	248	3&4	27	178
4&5	79	325	4&5	24	224
5&6	45	390	5&6	14	273
6&7	36	483	6&7	24	328
			7&8	10	383
			8&9	17	433

processing tasks as compared to line segmentation in [8].

For example, two words  in line 9 and 11, respectively, are correctly segmented by CLS algorithm but [8] is incapable of such initial segmentation, therefore, it necessitates to process these tasks later on.

Furthermore, red circled tasks in Fig. 20 (a) are still there to be resolved after initial segmentation by [8], yet the proposed algorithm resolves more than half of them encircled as blue shown in Fig. 20(b).

#### Algorithm 4 Decide Primary and Secondary Components

**Data:**  $T^k$  = containing information of connected components, Connected Component=CC, Secondary Component Height < Connected Comp Height  $CCH = \frac{1}{2}$  of the height of a text line  
**Result:** Updated Table  $T^k$  =, containing information regarding connected components and their types.

```

begin
  for each  $CC_i$  that overlaps with any other  $CC_j$  in  $T^k$ 
  do
    if  $CC_i$  is not diacritic and  $SCC_j$  is a diacritic
    then
      if  $CC_j$  already allocated to some primary  $CC_o$  then
        if  $CC_i$  completely overlaps  $CC_j$  OR
        centroids of  $CC_i$  and  $CC_j$  are closer than
         $CC_o$  and  $CC_j$  OR overlapping area of
         $CC_i$  and  $CC_j$  is more than  $CC_o$  and  $CC_j$ 
        then
          update  $CC_j$  as secondary component
          of  $CC_i$  in table  $T^k$ 
        else
          if  $CC_j$  already not allocated to some
          primary component then
            if  $CC_i$  partially overlaps  $CC_j$  OR
            height of  $CC_j < CCH$  then
              update  $CC_j$  as secondary
              component of  $CC_i$  in table  $T^k$ 
            else
              if height of  $CC_j < CCH$  then
                update  $CC_j$  as secondary component of
                 $CC_i$  in table  $T^k$ 
              else
                if height of  $CC_i > CCH$  and height
                of  $CC_j < CCH$  and  $CC_j$  is touching baseline
                then
                  update  $CC_j$  as a secondary component of
                   $CC_i$  in table  $T^k$ 
      return  $T^k$ 

```

A recent line segmentation algorithm [9] also used the zone based method as defined by [8] with additional processing of dilating for better estimation of zone size.

Fig. 21(a) represents the initial segmentation process and there is still need to resolve some overlapping issues later on. On the other hand, output of the proposed algorithm has perfectly segmented the page into text lines and there is no need for further processing as depicted in Fig. 21(b).

A state of the art contribution [29] for layout of Urdu and Arabic script that is actually an improvement of [28], which



**Algorithm 5** Allocate Secondary to Primary Components

**Data:**  $T^k$  = contains information of connected components and their types, Secondary Connected Component=SCC, Primary Connected Component=PCC, Secondary Component Height  $<$  Connected Comp Height  $CCH = \frac{1}{2}$  of the height of a text line

**Result:** List of ligatures  $L^k = \{Ligature_i^k\}_{i=1}^m$

**begin**

**for** each  $SCC_i$  of  $T^k$  **do**

**if**  $SCC_i$  is not touching baseline (Non conflicting  $SCC_i$ ) **then**

      Copy pixels of  $SCC_i$  at proper coordinates on its allotted  $PCC$

**else**

**if**  $SCC_i$  touching baseline and height of  $SCC_i < CCH$  **then**

        Declare  $SCC_i$  as not touching baseline in  $T^k$

        Copy  $SCC_i$  at proper coordinates to its allotted  $PCC$

**else**

        Declare  $SCC_i$  as  $PCC$  in  $T^k$

**for** each  $PCC_i$  of  $T^k$  **do**

    Save  $PCC_i$  as a  $\{Ligature_i^k\}$ ,  $i^{th}$  ligature of  $k^{th}$  sentence

**return**  $L^k = \{Ligature_i^k\}_{i=1}^m$ ,  $m$ =number of ligatures of  $S_k$

has depicted two very complex sample Urdu Nastaleeq pages as shown in Fig. 22 (a) and (c).

The complexity of the pages may be observed by more interlacing and merging of components of subsequent lines. The average intensity of initial split line positions are 46 and 14 for first and second page, respectively, as presented in Table 3. The highest intensity values at split positions pose maximum obstacles in drawing boundary, whereas, a straight line can demarcate a boundary between two text lines in case of minimum intensity value (i.e. 0). The proposed line segmentation algorithm is applied on these two complex pages, that extracts text lines accurately with only one diacritic out of place encircled red as shown in Fig. 22(b). Furthermore, some encircled dots/diacritics are misplaced by algorithm of [29], as depicted in Fig. 22 (e).

The proposed line segmentation algorithm is also applied on an Arabic language page provided in [29]. Our line segmentation algorithm misplaced only one diacritic, while line segmentation algorithm of [29] misplaced 5 diacritics as shown in Figs. 23(b) and 23(c), respectively. Lines are segmented by both algorithms with 100% accuracy.


**C. RESULTS OF THE PROPOSED LIGATURE SEGMENTATION ALGORITHM**

UPTI (Urdu Printed Text Images) data set [18] is used for checking the segmentation accuracy of the proposed segmentation algorithms. This data set contains images of 10063 sentences. Four degradation techniques that were proposed in [32] are applied, which include jitter, elastic elongation, threshold and sensitivity. Every degradation technique is applied with four different parameter values. Overall, UPTI contains 12 degradation versions of original 10063 sentence images. The sentence images of UPTI data set from undegraded version is segmented into ligatures as presented in Table 4.

There are total of 189584 ligatures as per split of ground truths of UPTI data set. Accuracy of the implemented algorithms is even better than the best ligature segmentation accuracy reported in [8] i.e. 99.02%.

UPTI data set contains incorrectly merged ligatures that badly affect the accuracy, because the corresponding ligatures in ground truths are not properly connected. Therefore, count of image ligatures reduces as compared to ligatures segmented from text lines of ground truth, resulting in low accuracy. Examples of these incorrectly connected dots, diacritics and ligatures are depicted in Fig. 24.

First  and  second part of Fig. 24 represent two ligatures connected incorrectly. Fourth part 

consists of all secondary components of third part  and represents six dots connected incorrectly. The overall size of these connected dots becomes more than the maximum threshold size of a secondary component, also it is touching the baseline. Therefore, ligature segmentation algorithm declares it as a primary component.

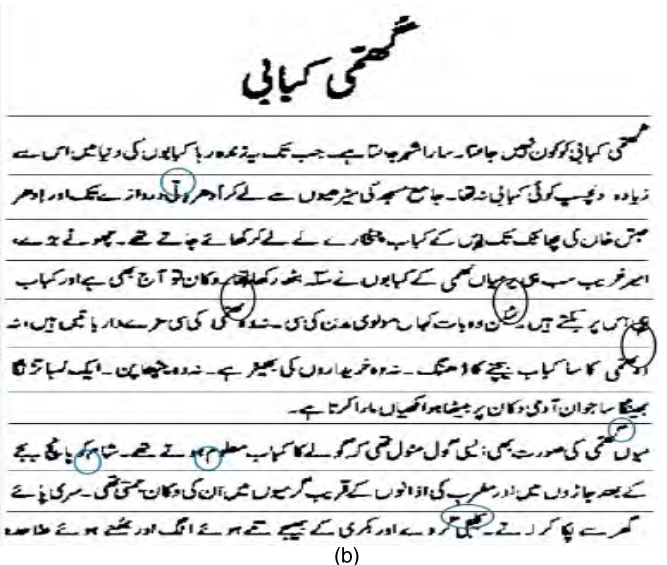
The proposed algorithm is tested for segmenting undegraded version of the UPTI data set. It is very interesting that the overall secondary components are less than the primary components. Further, the ratio of primary to secondary components is about 7:5. There are total of 189,584 ligatures as per split of ground truths of UPTI data set. Ground truths don't have any incorrect connectivity, so, in this way UPTI data set contains 189,584 ligatures and our algorithm extracts the ligatures as mentioned in Table 4.

Table 4 indicates that about 189 000 ligatures are successfully extracted from 10063 text lines consisting of above 332 000 connected components from UPTI data set. Furthermore, total of about 143 000 secondary components are allocated to approximately 189 000 primary ligatures with an accuracy rate of 99.80%.

Fig. 25 depicts the percentage statistics of ligature segmentation algorithm for four versions of UPTI data set. The primary components touching the baseline are approximately 93%. Similarly, the secondary components that are

**TABLE 4.** Statistics about components of UPTI data set.

	Primary components		Secondary components		Total components	Ligature Segmentation	Segmentation accuracy
	Touching baseline	Not touching baseline	Touching baseline	Not touching baseline			
UPTI data set undegraded	176353	12879	18801	124459	332492	189232	99.80%

**FIGURE 19.** Initial line segmentation by [8] and proposed line segmentation algorithm. (a) Line segmentation by [8] after segmentation of merged lines. (b) Initial line segmentation by proposed line segmentation algorithm.**FIGURE 20.** Final segmented text lines by [8] and proposed line segmentation algorithm.

not touching the baseline are approximately 87%, while the rest are not touching the baseline. Besides, every ligature consists of approximately 1.7 connected components. Table 5 represents the observed statistics by applying the proposed

ligature segmentation algorithms to the first sentence of UPTI data set.

Table 6 presents the results of segmentation algorithms and their accuracy along with the sizes of data sets used. The pri-

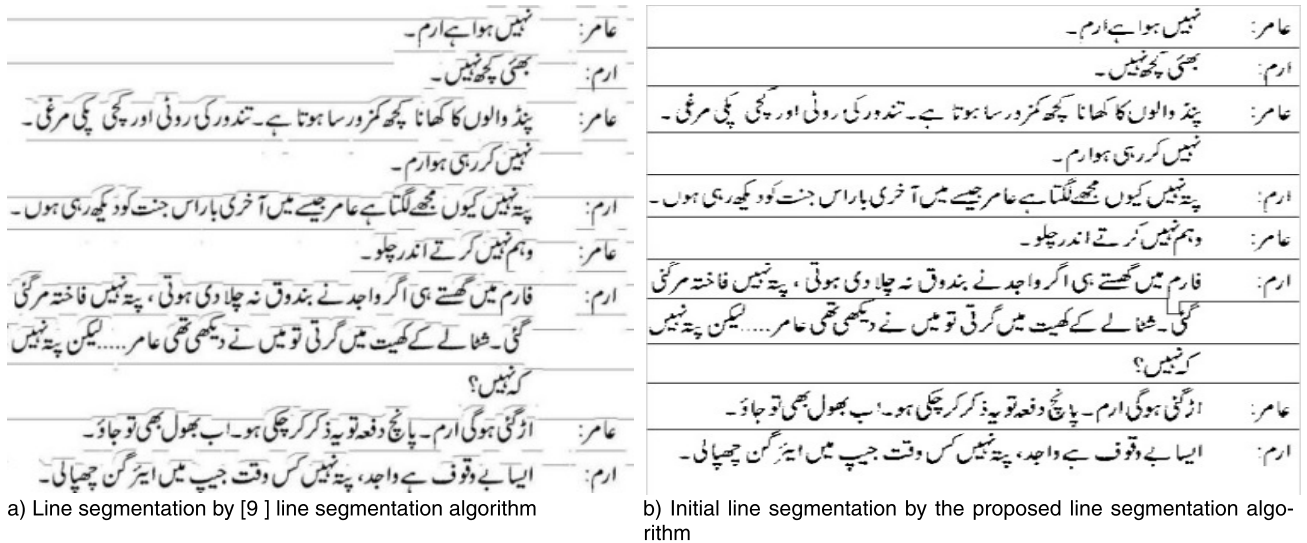


FIGURE 21. Comparison of line segmentations of [9] and the proposed line segmentation algorithm.



FIGURE 22. Urdu Nastaleeq pages taken from [29] and segmented by the proposed line segmentation algorithm.

TABLE 5. Statistics about components of first sentence of UPTI data set.

Primary comp. touching baseline		Secondary comp. touching baseline		Total comp.	Ligatures segmented
Yes	No	Yes	No		
21	6	4	20	51	27

many components that are touching baseline with size greater than the threshold size of a primary component are considered as primary components without confusion and rest are with confusion. Similarly, conflicting (with confusion) and non-conflicting (no confusion) secondary components are explained in Algorithm 5.

Javed and Hussain [7] used total 3655 ligatures, having 3375 correctly segmented ligatures with 92% accuracy, as presented in first row of Table 6. Din et al. [9] tested



FIGURE 24. Wrongly connected components of ligatures.

their algorithm on approximately 7364 ligatures with 99.49% accuracy. Lehal [8] tested his algorithm on 23 555 ligatures with 99.02% accuracy. Last row of Table 6 depicts the evaluation of our algorithm on first version of UPTI data set, as presented in Table 4. Also, 10 063 text lines consisting of more than 332 000 connected components from every subset of data set were extracted. Furthermore, total of about 143 000 secondary components were allocated to approximately 189 000 primary ligatures with the accuracy rate of 99.80%.

The best ligature segmentation algorithm [8], [9] are using six heuristics for segmentation into ligatures. Instead of heuristics, we are using quantitative values (width, height,



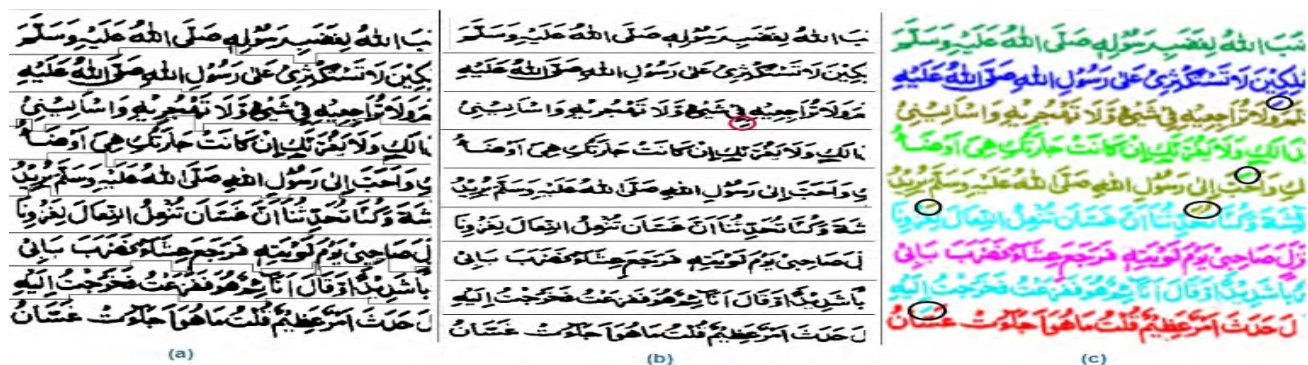


FIGURE 23. Arabic Nastaleeq page taken from [29] and segmented by proposed line segmentation algorithm.

TABLE 6. Comparison of Urdu language ligature segmentation algorithms.

Ligature segmentation algorithm	Primary components			Secondary components			Total comp.	Ligatures	Accuracy
	Total	No confusion	With confusion	Total	No confusion	With confusion			
Javed et.al[7]								Corr./total 3375/3655	92%
Din. et. al [9]								6811/7364	99.49%
Lehal [8]		23555		18886	17565	12805	42441	23555	99.02%
This paper	189232	176353	12879	143260	18801	124459	332492	189232	99.80%

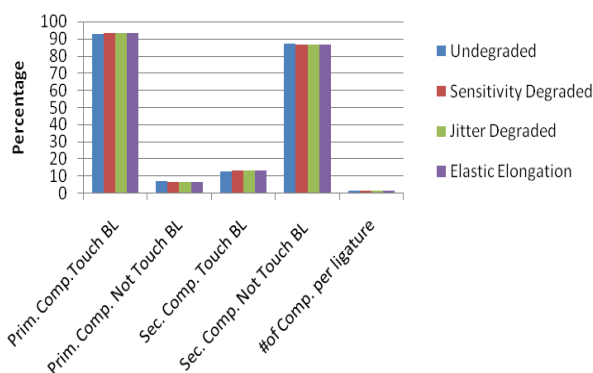


FIGURE 25. Percentage statistics about components of UPTI data set.

centroids and baseline touching) for segmenting lines into ligatures.

## V. CONCLUSION AND FUTURE WORK

The ongoing research work on line and ligature based Urdu Nastaleeq recognition systems motivated us to put forward more accurate segmentation algorithms for Urdu Nastaleeq document images. This proposal mainly introduces two algorithms for line and ligature segmentation of Nastaleeq text images. The proposed line segmentation algorithm places dots and diacritics more accurately as compared to [28], [29]. Prevailing work [8], [9] relied more on zonal information and heuristics for line and ligature segmentation, respectively. In our approach, we rely more on baseline for decision making

as opposed to zonal information for line segmentation. Also, proposed algorithm employs baseline, width, height, centroids, and overlapping quantitative information for ligature segmentation algorithms.

Furthermore, we tested our ligature segmentation algorithms on a very large data set with improved accuracy as presented in Table 4. On average, for UPTI data set, the proposed algorithm segmented 189 000 ligatures from 10 063 text lines with 332 000 connected components. A total of about 142 000 secondary components have been successfully allocated to more than 189 000 primary ligatures. The results of our line and ligature segmentation algorithms with accuracy of 99.17% and 99.80% respectively, are thus, better than the precision realized by the prevailing Urdu Nastaleeq segmentation algorithms.

This work can easily be extended for other Nastaleeq script based languages like Persian, Pashto, Siraiki, and Panjabi etc. Also, these algorithms can be modified for extraction of lines and ligatures from handwritten and scene Nastaleeq text. Segmentation of multicolumn Urdu newspapers text is one of the challenging task for future work.

## REFERENCES

- [1] A. Daud, W. Khan, and D. Che, "Urdu language processing: A survey," *Artif. Intell. Rev.*, vol. 47, no. 3, pp. 279–311, 2017.
- [2] S. Naz, K. Hayat, M. I. Razzak, M. W. Anwar, S. A. Madani, and S. U. Khan, "The optical character recognition of Urdu-like cursive scripts," *Pattern Recognit.*, vol. 47, no. 3, pp. 1229–1248, 2014.
- [3] G. S. Lehal and A. Rana, "Recognition of nastalique urdu ligatures," in *Proc. 4th Int. Workshop Multi-Lingual OCR*, 2013, p. 7.

- [4] U. Pal and A. Sarkar, "Recognition of printed urdu script," in *Proc. ICDAR*, 2003, pp. 1183–1187, 2003.
- [5] D. Satti and K. Saleem, "Complexities and implementation challenges in offline urdu nastaliq OCR," in *Proc. Conf. Lang. Technol.*, 2012, pp. 85–91.
- [6] S. Hussain, "Complexity of Asian writing systems: A case study of Nafees Nasta'leeq for urdu," in *Proc. 12th AMIC Annu. Conf. e-Worlds, Governments, Bus. Civil Soc., Asian Media Inf. Center*, Singapore, 2003.
- [7] S. T. Javed and S. Hussain, "Improving Nastaliq specific pre-recognition process for Urdu OCR," in *Proc. IEEE 13th Int. Multitopic Conf. (INMIC)*, Dec. 2009, pp. 1–6.
- [8] G. S. Lehal, "Ligature segmentation for urdu OCR," in *Proc. 12th Int. Conf. Document Anal. Recognit. (ICDAR)*, Aug. 2013, pp. 1130–1134.
- [9] I. U. Din, Z. Malik, I. Siddiqi, and S. Khalid, "Line and ligature segmentation in printed Urdu document images," *J. Appl. Environ. Biol. Sci.*, vol. 6, no. 3, pp. 114–120, 2016.
- [10] I. Shamsheer, Z. Ahmad, J. K. Orakzai, and A. Adnan, "OCR for printed Urdu script using feed forward neural network," in *Proc. World Acad. Sci., Eng. Technol.*, vol. 23, pp. 172–175, Jan. 2007.
- [11] I. K. Pathan and R. Ramteke, "Recognition of offline handwritten isolated urdu character," *Adv. Comput. Res.*, vol. 4, no. 1, pp. 117–121, 2012.
- [12] J. Tariq, U. Nauman, and M. U. Naru, "Softconverter: A novel approach to construct OCR for printed Urdu isolated characters," in *Proc. 2nd Int. Conf. Comput. Eng. Technol. (ICCET)*, vol. 3, Apr. 2010, p. V3-495.
- [13] Q. U. A. Akram, S. Hussain, and Z. Habib, "Font size independent ocr for noori nastaleeq," in *Proc. Graduate Colloquium Comput. Sci. (GCCS)*, vol. 1, Lahore, Pakistan, 2010.
- [14] K. Khan, R. Ullah, N. A. Khan, and K. Naveed, "Urdu character recognition using principal component analysis," *Int. J. Comput. Appl.*, vol. 60, p. 11, Dec. 2012.
- [15] Z. Ahmad, J. K. Orakzai, I. Shamsheer, and A. Adnan, "Urdu Nastaleeq optical character recognition," in *Proc. World Acad. Sci., Eng. Technol.*, vol. 26, 2007, pp. 249–252.
- [16] T. Nawaz, S. Naqvi, H. ur Rehman, and A. Faiz, "Optical character recognition system for Urdu (naskh font) using pattern matching technique," *Int. J. Image Process.*, vol. 3, no. 3, p. 92, 2009.
- [17] S. Hussain et al., "Nastaliq segmentation-based approach for urdu OCR," *Int. J. Document Anal. Recognit.*, vol. 18, no. 4, pp. 357–374, 2015.
- [18] N. Sabbour and F. Shafait, "A segmentation free approach to arabic and urdu OCR," in *Proc. DRR*, Feb. 2013, p. 8658.
- [19] S. T. Javed and S. Hussain, "Segmentation free nastaliq urdu OCR," in *Iberoamerican Congress on Pattern Recognition*. Turkey: Springer, 2013.
- [20] A. Ul-Hasan, S. B. Ahmed, F. Rashid, F. Shafait, and T. M. Breuel, "Offline printed urdu Nastaleeq script recognition with bidirectional LSTM networks," in *Proc. 12th Int. Conf. Document Anal. Recognit. (ICDAR)*, Aug. 2013, pp. 1061–1065.
- [21] S. Naz, A. I. Umar, R. Ahmad, S. B. Ahmed, S. H. Shirazi, and M. I. Razzak, "Urdu Nasta'liq text recognition system based on multi-dimensional recurrent neural network and statistical features," *Neural Comput. Appl.*, vol. 28, no. 2, pp. 1–13, 2015.
- [22] S. Naz et al., "Offline cursive Urdu-Nastaliq script recognition using multidimensional recurrent neural networks," *Neurocomputing*, vol. 177, pp. 228–241, Feb. 2016.
- [23] S. Naz, S. B. Ahmed, R. Ahmad, and M. I. Razzak, "Zoning features and 2DLSTM for urdu text-line recognition," *Proc. Comput. Sci.*, vol. 96, pp. 16–22, Oct. 2016.
- [24] I. Ahmad, X. Wang, R. Li, and S. Rasheed, "Offline urdu Nastaleeq optical character recognition based on stacked denoising autoencoder," *China Commun.*, vol. 14, no. 1, pp. 146–157, Jan. 2017.
- [25] H. Malik and M. A. Fahiem, "Segmentation of printed urdu scripts using structural features," in *Proc. 2nd Int. Conf. Vis. (VIZ)*, 2009, pp. 191–195.
- [26] K. S. Kumar, S. Kumar, and C. Jawahar, "On segmentation of documents in complex scripts," in *Proc. 9th Int. Conf. Document Anal. Recognit. (ICDAR)*, vol. 2, 2007, pp. 1243–1247.
- [27] T. M. Breuel, "Two geometric algorithms for layout analysis," in *International Workshop on Document Analysis Systems*. Berlin, Germany: Springer, 2002, pp. 188–199.
- [28] F. Shafait, A. Hasan, D. Keysers, and T. M. Breuel, "Layout analysis of urdu document images," in *Proc. 10th IEEE Int. Multitopic Conf. (INMIC)*, Islamabad, Pakistan, Dec. 2006, pp. 293–298.
- [29] S. S. Bukhari, F. Shafait, and T. M. Breuel, "High performance layout analysis of arabic and urdu document images," in *Proc. Int. Conf. Document Anal. Recognit. (ICDAR)*, Sep. 2011, pp. 1275–1279.
- [30] S. A. Husain, "A multi-tier holistic approach for urdu Nastaliq recognition," in *Proc. Int. Multi Topic Conf. Abstracts*, 2002, p. 84.
- [31] S. T. Javed, S. Hussain, A. Maqbool, S. Asloob, S. Jamil, and H. Moin, "Segmentation free nastaliq urdu OCR," *World Acad. Sci., Eng. Technol.*, vol. 46, pp. 456–461, Oct. 2010.
- [32] H. S. Baird, "Document image defect models," in *Structured Document Image Analysis*. Berlin, Germany: Springer, 1992, pp. 546–556.
- [33] S. S. Bukhari, F. Shafait, and T. M. Breuel, "Towards generic text-line extraction," in *Proc. 12th Int. Conf. Document Anal. Recognit. (ICDAR)*, Aug. 2013, pp. 748–752.
- [34] S. S. Bukhari, F. Shafait, and T. M. Breuel, "Text-line extraction using a convolution of isotropic Gaussian filter with a set of line filters," in *Proc. Document Anal. Recognit. (ICDAR)*, 2011, pp. 579–583.
- [35] S. S. Bukhari, T. M. Breuel, and F. Shafait, "Textline information extraction from grayscale camera-captured document images," in *Proc. 16th IEEE Int. Conf. Image Process. (ICIP)*, Nov. 2009, pp. 2013–2016.



deep learning, and algorithms.



He is a member of the Council of Chinese Information Processing Society and member of the Chinese Processing Committee of China Computer Federation.



ing Scholar with the Information Sciences Institute, University of Southern California, CA, USA. He is currently an Assistant Professor with the School of Computer Science, Beijing University of Posts and Telecommunications, and affiliated with the Engineering Research Center of Information Networks, Ministry of Education. His current research activities include neural information processing, multimedia information processing, and statistical machine learning.



**MANZOOR AHMED** received the B.E. and M.Phil. degrees from Pakistan, in 1996 and 2010, respectively, and the Ph.D. degree from the Beijing University of Posts and Telecommunications, China, in 2015. He is currently holds a post-doctoral position with the Department of Electronic Engineering, FIB Lab, Tsinghua University, China. His research interests include the image processing, game theoretic-based resource management in hierarchical heterogeneous net-

works, interference management in small cell networks, and 5G networks, D2D communication resource allocation and physical layer security and information security, Fog-RAN, C-RAN, and fog computing. He was a recipient of the Best Paper Award at the 2014 Game Nets Conference.



**RAHAT ULLAH** received the master's degree in electronics from the University of Peshawar, Pakistan, in 2007, and the Ph.D. degree from Beijing University of Posts and Telecommunications, China, in 2017. He was an Assistant Professor with Sarhad University, Peshawar, Pakistan, for six years. He joined the School of Physics and Optoelectronics Engineering, Nanjing University of Information Science and Technology, as an Associate Professor. His research interests include

all optical signal processing/communications, and machine learning.

• • •