Multimedia Tools and Applications (2023) 82:2393–2418
https://doi.org/10.1007/s11042-022-13243-x

Persian printed text line detection based on font size

Amirreza Fateh[1] · Mohsen Rezvani[1] · Alireza Tajary[1] · Mansoor Fateh[1]

Received: 26 December 2020 / Revised: 9 April 2021 / Accepted: 16 May 2022 /
Published online: 24 June 2022

Abstract
Text line segmentation is an essential step in the process of converting document images into text. In OCR systems, text line segmentation affects the character segmentation stage that has a direct effect on the recognition rate of the system. In scanned images, some lines are skew or curl, and the correct recognition of these lines is another challenge in this field. Also, in some languages like Persian, some of the words have diacritic. In this paper, we introduce a novel text line segmentation method based on the final font size for Persian printed document images to solve these problems. In this method, by finding a specific size, the Connected-Components in a line are glued together. To this end, the pre-processing step of the proposed method removes every small object from the input image using a de-noising method. In the next step, the method measures the diameter of each connected component (CC) in the image to detects the final font size. In the last step, the method finds all CCs that horizontally are in the same direction and then connects them. Due to the lack of a Persian OCR dataset, we created such a dataset. The experimental results are executed on this dataset, and the proposed method reached 99.3% accuracy. It is important to note that this dataset has some curved lines, which increases the challenges in the dataset.

Keywords Line detection · Font size · Persian printed · Connected component

1 Introduction

Recently, there is a substantial growth in the number of photo-to-text software, and many organizations are trying to digitize all their documents to get rid of their paper-based archived files. To this end, after scanning the documents, the scanned images are given to an image-to-text system to extract the text. Text-line detection usually uses for improving the accuracy of optical character recognition (OCR) techniques [9, 11, 26, 38, 43, 50]. Also, it is one of the pre-processing steps in document layout analysis [9, 27, 38], text restoration [22, 43], binarization [10], and camera-captured images [12, 17, 38, 53, 61].

One of the main requirements in designing text-line detection methods is having high accuracy in extracting lines with a low computational cost [30]. Image distortions like the

✉ Mansoor Fateh
mansoor_fateh@shahroodut.ac.ir

[1] Faculty of Computer Engineering, Shahrood University of Technology, Shahrood, Iran

blurring of images, image acquisition by the low-resolution camera, and the light of mobile camera flash are the major obstacle to achieve high accuracy in such methods [11, 26]. These distortions may happen in pages scanned by scanners or mobile phone cameras. The OCR methods try to remove these obstacles as much as possible in the pre-processing step [9, 26].

Another major problem that we face in the segmenting of a text line is non-linear warping. Warping decreases the readability of the text as well as reduces the OCR accuracy [26]. In the last decade, several methods have been proposed to overcome non-linear warping, which are known as the de-warping methods [8, 19, 57, 58]. These methods can divide into two categories: 1) 2D image processing techniques [19, 58], 2) 3D document shape restoration techniques [8, 57]. Several methods [19, 41, 58, 60, 64] as well as the proposed method fall into the first category. The second category needs image capture with some special camera; also, for displaying the document surface, the method needs to use a 3D shape model [26].

In addition to the problems mentioned above, in languages like Arabic and Persian, the letters divide into two groups: connectors and non-connectors. Each connector letter connects to the next letter in a word, which makes the segmentation more difficult. Also, in these languages, some words have diacritic. The existence of diacritic has a reverse effect on the accuracy in extracting lines [4]. In recent years, several text-line detection techniques proposed for different languages like English, Chinese, French, Italian, and German. On the other hand, there is still much work to be done for languages like Persian [2].

In this paper, we present a novel approach for text line detection. The method starts by binarizing the input image, then removes all small objects from it. In the next step, the method measures the diameter of each CC in the image and defines the final font size of the image. Using the final font size of the image as the searching length, the method finds the nearest horizontal neighbor of each CC, then connects all these CCs that are horizontally in the same direction, and extracts all text lines.

We make the following contributions in this paper. The proposed method works on three-channel RGB and grayscale images. The method is based on the size of the font extracted from the image, which plays an important role in extracting curved lines. We have also introduced a new dataset, which contains curved lines.

The remaining of the paper is organized as follows: a brief description of previous text-line detection approaches is presented in Section 2. The proposed method is described in Section 3. Performance evaluation and experimental results are given in Section 4, followed by conclusions in Section 5.

2 Related works

As we mentioned, most of the existing text-line segmentation techniques are mainly proposed for pre-processing steps of the document layout analysis, optical character recognition (OCR), text restoration, binarization and camera-captured images. Several text-line segmentation approaches have been proposed in the last decades. These approaches classified as techniques based on: 1) CCs based methods [30, 37, 38, 43], 2) project profile-based methods [1, 5, 16, 25, 33, 36, 54, 56, 63], 3) smearing-based methods [3, 4, 7, 23, 33, 44, 56], 4) bounding-based methods [56], 5) Hough transform-based methods [46], 6) combined methods [51, 56], 7) tree-based methods [45], and 8) deep learning-based methods [15, 34, 42]. These techniques are discussed in this section.

Mahmood and Srivastava [43] proposed a text line segmentation technique for Urdu typewritten text based on edge information of the CCs. This technique is tested over two benchmark datasets that collected, compiled and organized by themselves with an accuracy of 87.36% and 84.75% respectively.

H. I. Koo [38] introduced a text-line detection method for the camera-capture document images which developed by incorporating state estimation into the connected-components. They extracted connected-components with the maximally stable extremal region (MSER) method and then estimated the scales and orientations of connected-components after they segmented the text-lines.

Yandong Guo et al. [30] proposed a text line detection method based on estimated optimization cost for text line direction; in the first step, they extracted the CC from the image. Then they appraised the direction of the text line in several local regions and tried to optimize their estimation.

Koichi Kise et al. [37] proposed a text line segmentation method based on the Voronoi diagram based on the CC analysis. They use the Voronoi diagram to get the approximate coordinates of each line. The page segmentation process can be considered as selecting the appropriate sections of the line as the boundaries of the document components. Also, Lawrence O'Gorman [47] proposed a new text line segmentation method based on the nearest-neighborhood clustering of CCs extracted from the document image.

Projection profile is a technique for text-line segmentation, classified into two main types: 1) vertical projection profile and 2) horizontal projection profile [1, 5, 16, 25, 56, 63]. The horizontal projection profile finds the interline gaps between lines and considers it a separator between two consequent lines. This approach is used for printed-text, where there is no overlapping or touching between lines [16, 33, 36, 54, 56]. Also, the techniques based on the strip can deal with overlapping and touching text lines [25].

The smearing methods use for text-line segmentation; these methods are trying to segment the text area by making similar regions along with the text locations [4, 44]. This approach uses both types of text (handwritten texts and printed texts), but it has some weaknesses; it fails in case of overlapping or touching between lines [3, 4, 7, 23, 33, 56].

Another solution for text line segmentation is detecting the bounding box. First of all, the method generates the histogram of the image, then specifies lines with a lesser number of pixels, then tries to find the centroid of each line and determines the boundaries for each line [56].

The Run-Length Smoothing method is used in smearing [51, 56]. This approach, at first, smears the adjacent black pixels along the horizontal direction. If the distance of white space between black pixels becomes less than the predefined threshold, the white pixels turn black. The bounding boxes of the CCs surround text lines. Finally, using the Run-Length Smoothing method, the strength of the histogram will be increased [51].

One of the practical methods used in text-line segmentation is the Hough Transform (HT) [46]. Two of the most efficient methods of Hough Transform used for line segmentation are Standard Hough Transform (SHT) and Progressive Probabilistic Hough Transform (PPHT) [46]. Hough peaks are determined, and then, according to these peaks, the lines are extracted [4].

One of the tree-based segmentation methods is XY-Cut [45], where each page is located at the root of the tree and the final segmented areas are in the leaves of the tree. The method recursively divides the image into two smaller rectangular areas showing tree nodes.

Recently, researchers use deep learning techniques to segmenting text lines to reach more accuracy. For extracting the document area, Lyu Bing et al. [42] use ARU-Net [28] to segment the text line. ARU-Net is a deep learning model for text line segmentation in historical documents, which applies A-Net and RU-Net as base networks. In the extraction process, some text lines may miss. Lyu Bing et al. use LeNet, a simple deep learning method [42], to classify extracted articles and handwritten pictures.

Deep learning is one of the most effective techniques for text line segmentation, but to gain a high accuracy, it needs to be trained by very large datasets. Due to the lack of a Persian OCR dataset, we can not use these methods.

As mentioned before, one of the problems in the text line detection field is detecting the curve/skewed text lines. There are several approaches proposed in the literature to mitigate this problem. Hassan El Bahi et al. [18] proposed a new method to recognize the texts taken from smartphones. In the pre-processing step, the method detects all contours of the image, then by dilation technique removes the background of the image, finally by connecting the CCs in each line, segments the region into text line images.

Rituraj Soni et al. [55] introduced a new method for the classification of the text regions in natural scene images. Achieving this aim depends on choosing optimal-classifiers and optimal-features. Using the improved MSER method, their method detects the possible text regions; then, extracts eleven features from these regions. Using the CfsSubsetEval and BFS parameters of the Weka tool [32], the method chooses an optimal-feature set from these eleven features to discriminate the text and non-text components. In the end, with the help of the Weka tool on the ICDAR 2013 training set, the method train many classifiers using these optimal-features.

A hierarchical recursive text detection method is proposed in [59] to detect texts in complex scenes. Besides, to detect these kinds of texts in complex scenes, [14] proposed an Adaptive Convolution and Path Enhancement Pyramid Network (ACPEPNet), which can more accurately locate the text instances with arbitrary shapes. In [31], The method combines the saliency model with the text detection approach in a natural scene to generate text saliency. The proposed method in [49] detects text regions from camera captured images by using the Fuzzy Distance Transform that is based on an adaptive stroke filter. Finally, In [35], a random forest classifier is used to recognize the newspaper texts. The inputs of the random forest classifier are the features of characters extracted by different kinds of feature extraction techniques.

OCRopus is a neural network-based approach that was first introduced in 2008 and presents good pre-processing in OCR [6]. This method has given several updates to date. The latest update of this method in 2017, which is available in [39], includes the following sections: line detection, optical character recognition using LSTM, and using statistical linguistic models.

The specific type of OCR software that we employed in our tests is an open-source OCR program called Kraken, developed by Benjamin Kiessling at Leipzig University's Alexander von Humboldt Chair for Digital Humanities. Kraken detects each line, then uses a neural network to recognize letters in each line. The latest Kraken update in the line recognition field was released in 2021, which is available in [40].

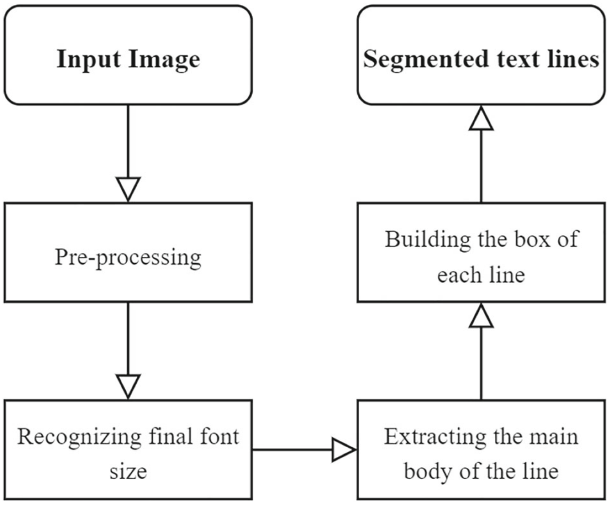

Input Image
Segmented text lines
Pre-processing
Building the box of each line
Recognizing final font size
Extracting the main body of the line

Fig. 1 Text line segmentation process

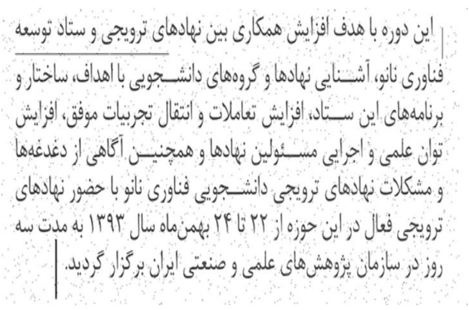
3 Proposed approach

In this section, we describe our text line segmentation method. Our proposed method has sub-processes of pre-processing, recognizing the final font size, extracting the main body of the line, and building the box of each line. These sub-processes are shown in Fig. 1. The proposed method removes noises and excess lines from the image in the "pre-processing" step. The proposed method has a bottom-up design. In this way, in the "recognizing final font size" step, it first extracts all CCs (CCs) of the image, then calculates the diameter of each of them, then the final font size calculates according to the definition. In the "extracting the main body of the line" step, Using the final font size of the image extracted from the previous step, the proposed method performs a search on the horizon axis around each of the CCs. If any other CCs are in that radius, the method connects the two neighboring CCs, thus extracting the main body of each line. In the last step, the proposed method tries to find any point or diacritic that does not stick to the main body of the line, and by using the projection technique, extracts the whole line.

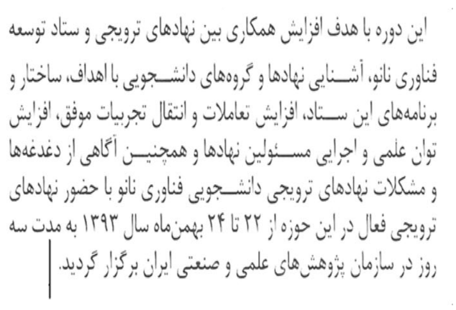
3.1 Pre-processing

Pre-processing is the first step in line segmentation. Algorithm 1 shows the steps of producing a clean image that facilitates the processing of the next steps. In this step, firstly, our method converts the 3-channel input image shown in Fig. 2a into a grayscale channel. It then obtains a threshold for the image to converts the grayscale image into binary. This threshold is calculated by the Otsu method [48]. Secondly, our method removes the salt-and-pepper noises from the image [29]. Initial labeling and measuring are the other steps of pre-processing.

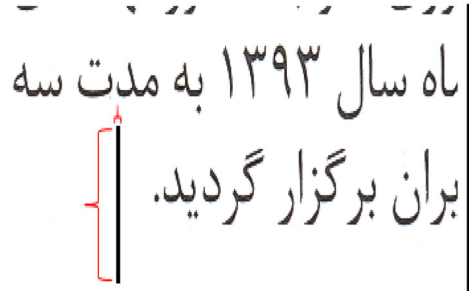
For de-noising, the method performs median filtering of the input image in two dimensions [24]. As shown in Fig. 2b, each output pixel contains the median value in a 3-by-3 neighborhood around the corresponding pixel in the input image. For initial labeling and measuring, the method tags every CC with a separate label to become recognizable from


این دوره با هدف افزایش همکاری بین نهادهای ترویجی و ستاد توسعه
فناوری نانو، آشـــنایی نهادها و گروه‌های دانشـــجویی با اهداف، ساختار و
برنامه‌های این ســـتاد، افزایش تعاملات و انتقال تجربیات موفق، افزایش
توان علمی و اجرایی مســـئولین نهادها و همچنیـــن آگاهی از دغدغه‌ها
و مشکلات نهادهای ترویجی دانشـــجویی فناوری نانو با حضور نهادهای
ترویجی فعال در این حوزه از ۲۲ تا ۲۴ بهمن‌ماه سال ۱۳۹۳ به مدت سه
روز در سازمان پژوهش‌های علمی و صنعتی ایران برگزار گردید.

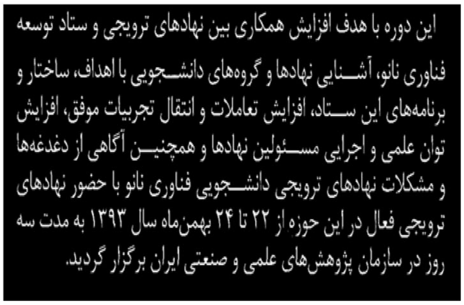
(a) Preprocessing input image.


این دوره با هدف افزایش همکاری بین نهادهای ترویجی و ستاد توسعه
فناوری نانو، آشـــنایی نهادها و گروه‌های دانشـــجویی با اهداف، ساختار و
برنامه‌های این ســـتاد، افزایش تعاملات و انتقال تجربیات موفق، افزایش
توان علمی و اجرایی مســـئولین نهادها و همچنیـــن آگاهی از دغدغه‌ها
و مشکلات نهادهای ترویجی دانشـــجویی فناوری نانو با حضور نهادهای
ترویجی فعال در این حوزه از ۲۲ تا ۲۴ بهمن‌ماه سال ۱۳۹۳ به مدت سه
روز در سازمان پژوهش‌های علمی و صنعتی ایران برگزار گردید.

(b) Output of median filtering.


اه سال ۱۳۹۳ به مدت سه
ران برگزار گردید.

(c) Input ofinitial labeling.

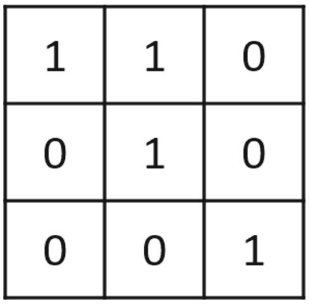

این دوره با هدف افزایش همکاری بین نهادهای ترویجی و ستاد توسعه
فناوری نانو، آشـــنایی نهادها و گروه‌های دانشـــجویی با اهداف، ساختار و
برنامه‌های این ســـتاد، افزایش تعاملات و انتقال تجربیات موفق، افزایش
توان علمی و اجرایی مســـئولین نهادها و همچنیـــن آگاهی از دغدغه‌ها
و مشکلات نهادهای ترویجی دانشـــجویی فناوری نانو با حضور نهادهای
ترویجی فعال در این حوزه از ۲۲ تا ۲۴ بهمن‌ماه سال ۱۳۹۳ به مدت سه
روز در سازمان پژوهش‌های علمی و صنعتی ایران برگزار گردید.

(d) Output of preprocessing phase.

Fig. 2 The pre-processing

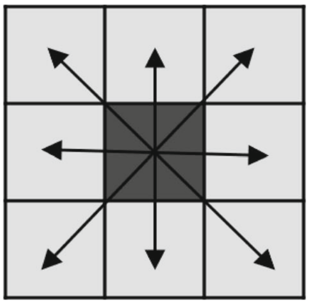
other CCs. To separate CCs from each other and assign a label to each, we used the MATLAB function bwlabel. As shown in Fig. 3, in this method, if the value of two adjacent pixels is one and these two pixels connect along the horizontal, vertical, or diagonal direction, these two pixels are part of the same object.

In the next step, the algorithm measures some properties of each label. These properties are the length (in pixels) of the major axis and the minor axis of the CC (label). The major Axis is the longest diameter of the CC. It goes from one side of the CC, through the center, to the other side, at the widest part of the CC. The minor axis is the shortest diameter (at

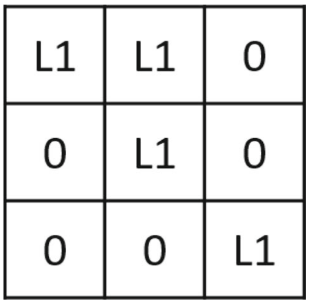


1	1	0
0	1	0
0	0	1

(a) Value of Pixels.



(b) Using MATLAB bwlabel.



L1	L1	0
0	L1	0
0	0	L1

(c) Labeling.

Figure 3: Using bwlabel

Fig. 3 Using bwlabel

the narrowest part of the CC). Finally, as shown in Fig. 2c, the method removes some labels that have this condition:

$$\frac{MajorAxisLength}{MinorAxisLength} > \varepsilon \tag{1}$$

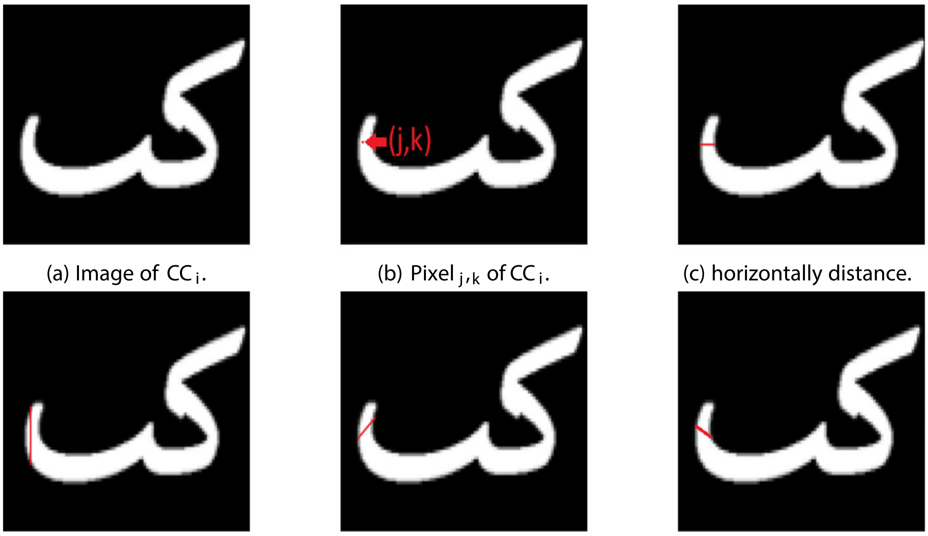
By using this condition, the method measures the diameter of the label. If *MajorAxisLength* is much bigger than *MinorAxisLength*, the method removes that label. The resulting image of the pre-processing phase is shown in Fig. 2d.

Algorithm 1 Pre-processing steps.

```
Result: The clean image
Function Binary($Input - image$):
    Input-image $\xrightarrow{\text{rgbTogray}}$ grayscale-image;
    grayscale-image $\xrightarrow{\text{OtsuMethod}}$ threshold;
    threshold, grayscale-image $\xrightarrow{\text{grayTobinary}}$ binary-image;
    return binary-image;
Function
median-filter($binary - image, Neighborhood - size : [M, M]$):
    for $i=1$:Number of binary image pixels do
        Value of pixel(i) = median value in a M-by-M neighborhood around pixel(i)
    end
    return modified-binary-image;
Function
Initial-labeling($modified - binary - image, epsilon - value : \varepsilon$):
    Initial labeling to each CC;
    Enclosing each Label (or CC) by ellipse;
    measuring the MajorAxisLength and MinorAxisLength of each ellipse;
    for $i=1$:Number of labels do
        if $\frac{MajorAxisLength(i)}{MinorAxisLength(i)} > \varepsilon$ or MajorAxisLength(i) == 1 or MinorAxisLength(i)
        == 1 then
            remove label(i)
        end
    end
    return clean-image;
Function Main:
    binary-image=Binary($Input - image$);
    modified-binary-image=median-filter($binary - image$, [3, 3]);
    clean-image=Initial-labeling($modified - binary - image$, 15);
```

3.2 Final font size recognition

This phase aims to find the final font size of the image. The method needs the final font size for connecting the CCs that are horizontally in the same direction.



(a) Image of CC$_i$. (b) Pixel$_{j,k}$ of CC$_i$. (c) horizontally distance.

(d) vertically distance. (e) obliquely distance 1. (f) obliquely distance 2.

Fig. 4 CC's diameter calculation

3.2.1 CC's diameter calculation

Our method starts by measuring the diameters of all CC_i's pixels (*where:* $0 \leq i \leq$ *Number Of CCs*). Consider Fig. 4a as the image of CC_i. In each pixel of CC_i, for example, the $pixel_{j,k}$ shown in Fig. 4b, the method measures the distance between two edges of CC_i at four different angles: horizontally (Fig. 4c), vertically (Fig. 4d), and obliquely in two directions (Fig. 4e and f). The Euclidean distance was used to calculate each of these distances. The Euclidean distance between two points like p and q is shown in (2).

$$d(p, q) = \sqrt{(q_x - p_x)^2 + (q_y - p_y)^2} \tag{2}$$

Filters such as canny filters [13] can be used to extract CC edges. However, there is no need to extract edges because the pixels' values are binary. Therefore, any pixel that has at least one black pixel in its neighborhood is an edge. By doing this, the speed of the algorithm increases.

Then it sets the minimum distance as the CC_i's diameter in that pixel. This is according to the definition of CC_i's diameter in this paper. At any point of CC_i, the closest distance between the two edges considered being the CC_i's diameter in that pixel. After completing this step, the diameter of the CC_i in all its pixels is specified. Then the diameter with the most repetitions among the CC_i's pixels is introduced as the final CC_i's diameter. Finally, the diameter value of all CC_i's pixels changes to this diameter value. The proposed method chooses the largest number of repetitions of the diameter, as the final diameter. For example, consider the CC_i pixels that are visible with the red box around them in Fig. 5. Table 1 shows the diameters extracted from these pixels. As shown in Table 1, most parts of the CC_i have the same diameter. Algorithm 2 summarizes this step.

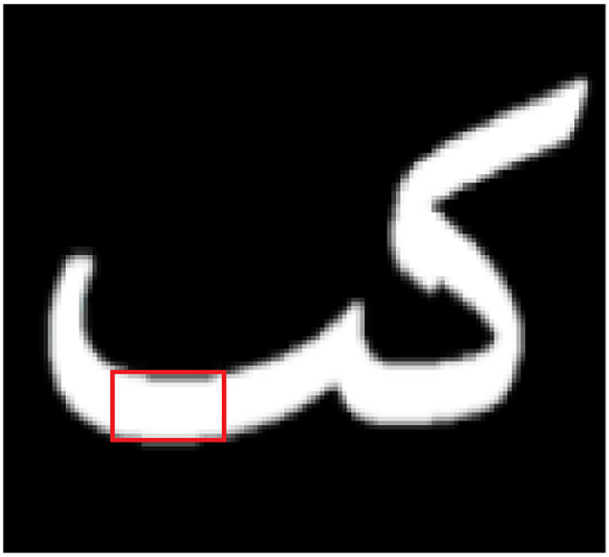


Fig. 5 Pixels' diameter

Algorithm 2 CC's diameter calculation steps.

```
for i=1:Number of CCs do
    foreach pixel of $CC_i$ do
        HD= horizontal distance between two edges of $CC_i$;
        VD= vertical distance between two edges of $CC_i$;
        OD1, OD2= oblique distance between two edges of $CC_i$;
        $CC_i$'s diameter of the pixel= minimum(HD,VD,OD1, OD2);
    end
    final $CC_i$'s diameter= the largest number of repetitions of the diameter among the $CC_i$;
    foreach pixel of $CC_i$ do
        $CC_i$'s diameter of the pixel=final $CC_i$'s diameter;
    end
end
```

3.2.2 Final font size

In the next step, the method gets all the different CC's diameters with their number of repetitions. The method chooses the diameter with the most number of repetitions as the first font size. To improve method's performance, we need to remove all points and diacritics before calculating the number of repetitions. Otherwise, due to the existence of these objects in documents, the algorithm may make errors in its calculations, and the final font size may extract based on the diameter of these points and diacritics.

For removing all points and diacritics, we used the MATLAB function bwareaopen. But before using this function, the algorithm needs to extract the number of pixels per CC. If the number of pixels is less than a predetermined value, that CC is removed. This function

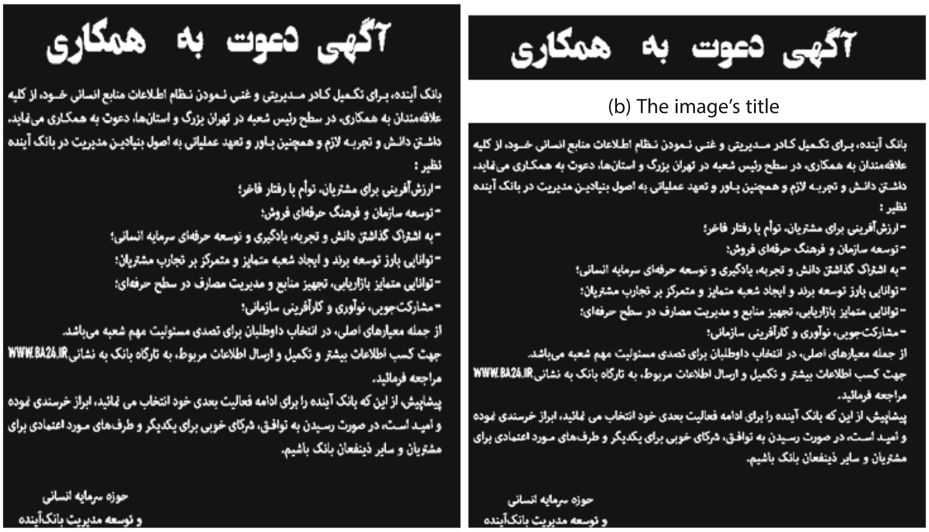
Table 1 Some pixel's diameter in CC_i

3	2.8	4	4	2.8	3
3	3	4	4.2	3	3
3	3	4	4.2	3	3
3	3	2	2	3	2.8

is used to remove small objects from binary images. Therefore, first, the proposed method finds the largest number of repetitions of the diameter in all CC_is, then calculates the number of pixels of each CC_i. Finally, the proposed method uses the bwareaopen function to remove CCs whose number of pixels is less than ten times the diameter with the most repetition. This ten times the largest number of repetitions of the diameter in CC_i has been achieved with many trial and error on different font types in various sizes and styles. By removing a CC, the proposed method removes CC's diameters from the diameters table. After eliminating all points and diacritics, the method sorts the remaining diameters on the image by number, then choosing the diameter with the greatest number of repetitions as the second font size.

For removing all points and diacritics, we used we used the MATLAB function bwareaopen. First, the method finds the largest number of repetitions of the diameter in CC_i, then calculates the number of pixels of the CC_i. Finally, the method removes CC_i's diameters from diameter's table if the number of pixels of the CC_i is less than ten times the largest number of repetitions of the diameter in CC_i. This ten times the largest number of repetitions of the diameter in CC_i has been achieved with a lot of trial and error on different font types in different sizes and styles. After removing all points and diacritics, the method sorts the remaining diameters on the image by number, then choosing the diameter with the most number of repetitions as the second font size.

Sometimes the font-size of the title is extremely big (Fig. 6a), and the method segments the title lines incorrectly. To solve this problem, first, the method separates the titles and


آگهی دعوت به همکاری

بانک آینده، برای تکمیل کادر مدیریتی و غنی نمودن نظام اطلاعات منابع انسانی خود، از کلیه علاقه‌مندان به همکاری، در سطح رئیس شعبه در تهران بزرگ و استان‌ها، دعوت به همکاری می‌نماید. داشتن دانش و تجربه لازم و همچنین باور و تعهد عملیاتی به اصول بنیادین مدیریت در بانک آینده نظیر :

- ارزش‌آفرینی برای مشتریان، توأم با رفتار فاخر؛
- توسعه سازمان و فرهنگ حرفه‌ای فروش؛
- به اشتراک گذاشتن دانش و تجربه، یادگیری و توسعه حرفه‌ای سرمایه انسانی؛
- توانایی بارز توسعه برند و ایجاد شعبه متمایز و متمرکز بر تجارب مشتریان؛
- توانایی متمایز بازاریابی، تجهیز منابع و مدیریت مصارف در سطح حرفه‌ای؛
- مشارکت‌جویی، نوآوری و کارآفرینی سازمانی؛

از جمله معیارهای اصلی، در انتخاب داوطلبان برای تصدی مسئولیت مهم شعبه می‌باشد.
جهت کسب اطلاعات بیشتر و تکمیل و ارسال اطلاعات مربوط، به تارگاه بانک به نشانی WWW.BA24.IR مراجعه فرمائید.
پیشاپیش، از این که بانک آینده را برای ادامه فعالیت بعدی خود انتخاب می نمائید، ابراز خرسندی نموده و امید است، در صورت رسیدن به توافق، شرکای خوبی برای یکدیگر و طرف‌های مورد اعتمادی برای مشتریان و سایر ذینفعان بانک باشیم.

حوزه سرمایه انسانی
و توسعه مدیریت بانک‌آینده

(a) Input-image (b) The image's title (c) The image's body

Fig. 6 Image with big title

saves them as a new image. Thus, we will have two images with different font sizes. As shown in Algorithm 3, the proposed method finds all unique diameters after removing all points and diacritics. If any of the unique diameters is more than twice the second font size, the method detects the title in the image and chooses the second font size as the body's font size (the final font size of the body image). As a result, any diameter after this distance, considered as the diameter of the title. The method considers the most repeated value of remaining unique diameters as the title's font size (the final font size of the title image). As shown in Fig. 6b, the CCs whose their diameter is in the range of the body text diameter, are placed in one image, and the rest of the CCs, are placed in another image (Fig. 6c).

The important point in the proposed method is to use the original image in the next step. This means, after removing the small objects and calculating the final font size, the remaining steps will use the original image, which also has all of the small objects.

Algorithm 3 Final font size recognition steps.

```
Function
final-font-sizes($UDiameters$, $UDiametersNum$, $SecondFontSize$):
    $BodyFontSize = SecondFontSize$;
    for i=1:Number of $UDiameters$ do
        if $UDiameter_i > (2 \times SecondFontSize)$ then
            *The image has a big title*;
            $TitleFontSize$ = The most number of repetition of the remaining
            $UDiameter$
        end
    end
    return $BodyFontSize$, $TitleFontSize$;
Function Main():
    FirstFontSize ⇐ Choosing the diameter with the most number of repetition;
    for i=1:Number of CCs do
        if Number of $CC_i$'s pixels $< (10 \times FirstFontSize)$ then
            removing $CC_i$ by using bwareaopen ($CC_i$'s Label = background Label);
            removing $CC_i$'s diameters from diameter's table;
        end
    end
    UD = Sorting unique diameters;
    UDNum = Calculating the number of repetitions of unique diameters;
    SecondFontSize ⇐ choosing the diameter with the most number of repetition;
    BodyFontSize, TitleFontSize =
     final-font-sizes(UD,UDNum,SecondFontSize);
    Title Image = [ ];
    Body Image = [ ];
    for i=1:Number of CCs do
        if $CC_i$'s diameter $< (2 \times BodyFontSize)$ then
            Body Image ⇐ $CC_i$
        else
            Title Image ⇐ $CC_i$
        end
    end
```

3.3 Extracting the main body of the line

The method has to connect any CCs that have been horizontally in the same direction to get the approximate lines' locations of the image. To reach this goal, every CC in the image tries to find the nearest horizontal neighbor by using a searching method in the horizontal direction. In this step, the line segmentation method uses the final font size as an input parameter and searches in the horizontal direction in the specified length to find another CC, if the searching method finds any white pixel that belongs to another CC, it connects these two CCs. The searching length depends on the size of the final font size. We set an adaptive coefficient for controlling the searching length. Using this adaptive coefficient and multiplying it by the final font size, the best search length is obtained.

In the next step, we used one of the morphological structuring elements called $strel$. Its basic syntax is:

$$se = strel(Shape, parameters) \tag{3}$$

Which we used one of the special types:

$$se = strel('line', Len, Deg) \tag{4}$$

This type of strel, creates a flat, linear structuring element, where Len specifies the length, and Deg specifies the angle (in degrees) of the line, as measured in the counterclockwise direction from the horizontal axis. Finally, with another morphological operation called closing, it connects CCs within the search radius of each other. And the lines are detected correctly. Algorithm 4 shows how to calculate this adaptive coefficient.

Algorithm 4 calculating the best searching length steps.

```
if $final - font - size < 50$ then
    $A = \frac{60 - final - font - size}{10}$
else
    $A = 1$
end
searching-length = A × final-font-size
se = strel ('line', searching-length, 0)
Body/Title image = imclose(Body/Title image, se)
```

3.4 Building the box of each line

Despite implementing all the previous steps to segment lines correctly, the method detects some of the up and down word's points as a new label (Fig. 7a). To handle this problem, we had to describe the last part of our proposed method in this step. Firstly, the method finds min(x), min(y), max(x) and max(y) for every label. As shown in Fig. 7b, if the position of the point's label is near to the main label, the method connects these labels. Otherwise, the point's label will be considered as another CC.

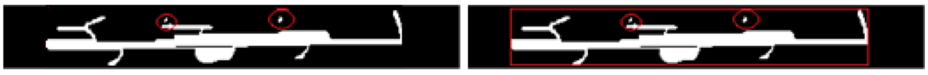

(a) Disconnected points (b) Bounding box

Fig. 7 Bounding Box steps

3.5 Time complexity

In Algorithm 1, the time complexity of the first two functions is in $O(N)$, in which N is the number of pixels in the image. Also, the time complexity of the third function is in $O(N + w)$, in which w is the number of labels in the image. And because N is so larger than W, we considered it as $O(N)$. Therefore, the time complexity of Algorithm 1 is equal to $O(N)$. In Algorithm 2, the time complexity is in $O(N_w^2)$, in which N_w is the number of pixels labeled in the image. In Algorithm 3, the time complexity both choosing the diameter with the most number of repetitions, and calculating the number of repetitions of unique diameters are in $O(N)$. Both for loops in the Main function are in $O(w)$. The final-font-sizes function is in $O(d)$, in which d is the number of unique diameters in the image. Therefore, the time complexity of Algorithm 3 is equal to $O(N)$. Finally, in Algorithm 4, the time complexity is in $O(N)$.

4 Results and evaluation

In this section, we explain and discuss the experimental results. We also provide information on how to make the dataset and the contents of them. Finally, we compare the results of the proposed method with related works on our dataset. To validate the effectiveness of the proposed system, we have conducted experiments with three different datasets. We have used our own system to implement and evaluate the proposed method. The CPU of our system is Intel(R)-Core(TM) i7-9750H, and our RAM is 16 GB.

4.1 Datasets

Due to the lack of a Persian OCR dataset, we made a standard dataset. The generated dataset is publicly available. We collected the documents' text of our dataset from various sources like newspapers, magazines, and books. The collected text is made in different sizes, fonts, and styles and saved as images in png format. Some of the words in these documents have diacritic.

To create datasets in different fonts, styles, and sizes from a text, we have used the python-docx library. Using this library, you can easily produce text in various sizes, styles, and fonts. The latest version of this library is for 2019. python-docx saves input text as a Microsoft Word file. For more details on this library, refer to the documentation provided for the python-docx library. Finally, each page of the Microsoft Word file will be saved as an image in png format. The size of each page is 1653×2339. This dataset is available in [21].

Our advantage over other existing datasets is the presence of different font types in various sizes and styles of Persian in a relatively significant number of lines. We have created images from six famous Persian font types in three styles and six different sizes.

We used six distinct font types to test our proposed method, including *Arial, Nazanin, Ziba, IranNastaliq, Tahoma,* and *XB-Niloofar*. These font types are common in writing magazines, books, and newspapers. We used all three types of bold, italic, and regular fonts in our dataset. We have also used font sizes 8, 10, 14, 18, and 22 in our dataset. Also, the total number of lines in this dataset is more than 24,000 lines. Table 2 shows a summary of the dataset information.

To further evaluate the performance of the proposed method, we used two more datasets to test the proposed method. The first dataset is in Arabic. This dataset which is called the Arabic OCR dataset is available in [62]. The images in this dataset are in different sizes.

Table 2 A statistic of the font types and sizes used in building our data-set

Font style	Font type	#of lines
Bold	Arial	1421
	Nazanin	1441
	Ziba	1231
	IranNastaliq	793
	Tahoma	1873
	XB-Niloofar	1460
Italic	Arial	1402
	Nazanin	1449
	Ziba	1227
	IranNastaliq	791
	Tahoma	1685
	XB-Niloofar	1434
Regular	Arial	1402
	Nazanin	1449
	Ziba	1227
	IranNastaliq	791
	Tahoma	1685
	XB-Niloofar	1434

Arabic OCR dataset aims to solve a more straightforward problem of OCR with images that contain only Arabic characters. Figure 8 shows an example of the images in this dataset.

To evaluate the performance of the proposed method, we randomly selected 50 images from this dataset and tested the proposed method on these images. The images in this dataset are single columns lines, but lines have a small amount of rotation, which increases the complexity of the dataset.

We prepared the second database ourselves. The images in this dataset are provided by the HP Scanjet 4890 scanner with default settings. We have scanned 80 images from various sources such as magazines, newspapers, and books with this scanner, which is available in [20]. This dataset's images have more complexity than the images of the previous two datasets. One of the most considerable complexities of the images in this dataset is the pages' rotation, which creates skew lines. As shown in Fig. 9, which is one of the images in this dataset, the images in this dataset have rotation, making line recognition difficult.

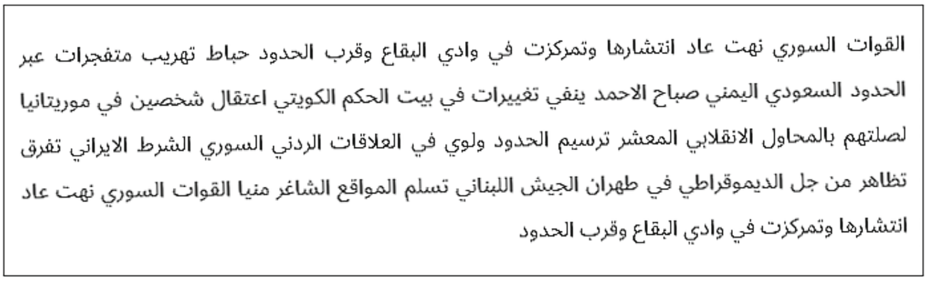
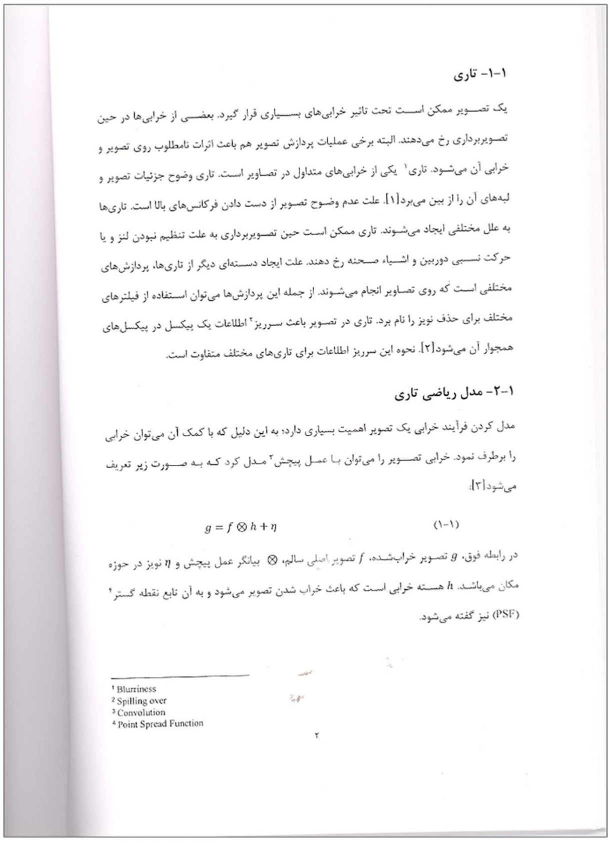

القوات السوري نهت عاد انتشارها وتمركزت في وادي البقاع وقرب الحدود حباط تهريب متفجرات عبر
الحدود السعودي اليمني صباح الاحمد ينفي تغييرات في بيت الحكم الكويتي اعتقال شخصين في موريتانيا
لصلتهم بالمحاول الانقلابي المعشر ترسيم الحدود ولوي في العلاقات الردني السوري الشرط الايراني تفرق
تظاهر من جل الديموقراطي في طهران الجيش اللبناني تسلم المواقع الشاغر منيا القوات السوري نهت عاد
انتشارها وتمركزت في وادي البقاع وقرب الحدود

Fig. 8 Sample of Arabic OCR dataset



۱-۱- تاری

یک تصـــویر ممکن اســـت تحت تاثیر خرابی‌های بســـیاری قرار گیرد. بعضـــی از خرابی‌ها در حین تصـویربرداری رخ می‌دهند. البته برخی عملیات پردازش تصویر هم باعث اثرات نامطلوب روی تصویر و خرابی آن می‌شـود. تاری[۱] یکی از خرابی‌های متداول در تصـاویر اسـت. تاری وضوح جزئیات تصویر و لبه‌های آن را از بین می‌برد[۱]. علت عدم وضـوح تصـویر از دست دادن فرکانس‌های بالا است. تاری‌ها به علل مختلفی ایجاد می‌شـوند. تاری ممکن اسـت حین تصـویربرداری به علت تنظیم نبودن لنز و یا حرکت نسـبی دوربین و اشـیاء صـحنه رخ دهند. علت ایجاد دسـته‌ای دیگر از تاری‌ها، پردازش‌های مختلفی اسـت که روی تصـاویر انجام می‌شـوند. از جمله این پردازش‌ها می‌توان اسـتفاده از فیلترهای مختلف برای حذف نویز را نام برد. تاری در تصـویر باعث سـرریز[۲] اطلاعات یک پیکسل در پیکسل‌های همجوار آن می‌شود[۲]. نحوه این سرریز اطلاعات برای تاری‌های مختلف متفاوت است.

۱-۲- مدل ریاضی تاری

مدل کردن فرآیند خرابی یک تصویر اهمیت بسیاری دارد؛ به این دلیل که با کمک آن می‌توان خرابی را برطرف نمود. خرابی تصـــویر را می‌توان بـا عمـل پیچش[۳] مـدل کرد کـه بـه صـــورت زیر تعریف می‌شود[۳]:

$$g = f \otimes h + \eta \qquad (۱-۱)$$

در رابطه فوق، g تصـویر خراب‌شـده، f تصویر اصلی سالم، $\otimes$ بیانگر عمل پیچش و η نویز در حوزه مکان می‌باشـد. h هسـته خرابی است که باعث خراب شدن تصویر می‌شود و به آن تابع نقطه گستر[۴] (PSF) نیز گفته می‌شود.

[1] Blurriness
[2] Spilling over
[3] Convolution
[4] Point Spread Function

۲

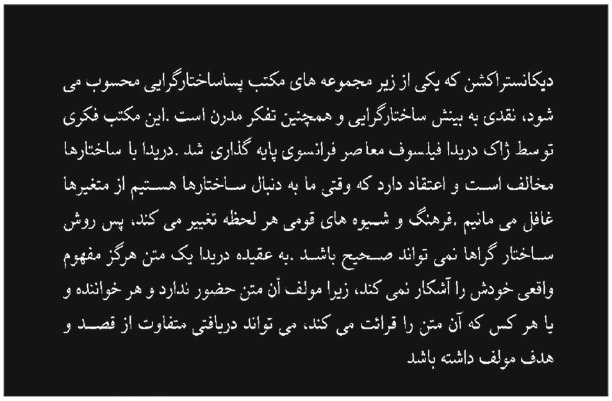
Fig. 9 Sample of dataset 3

Images' quality in dataset 3 is 300, 200, or 120 dpi. The maximum number of lines in the images of this dataset is 35 lines. Some images in this database were rotated up to about 10 degrees, and some other pages were curved, which increase the complexity of the dataset.

4.2 Parameter tuning

In the previous section, we defined some parameters, coefficients, and constants with specific values. In this subsection, we will explain in detail how to obtain these values. We said before the proposed method removes CCs which, the number of pixels is less than ten times the diameter with the most repetition. This Coefficient value has been achieved with many trial and error.

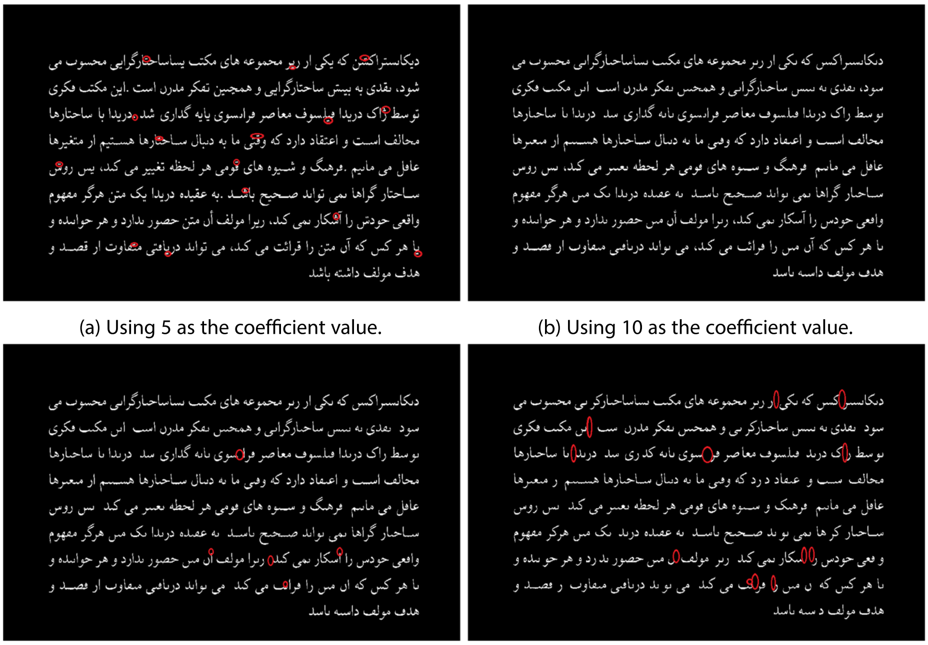
We obtained the best value for this coefficient among several values of 5, 10, 15, and 20 by performing a test on 50 images of the first dataset. Figure 10 shows part of a page that its text is in XB-Niloofar font type, regular font style, and font size 18. Figure 11 shows the result of using each of these coefficients in Fig. 10. As shown in Fig. 11a, some points are not omitted if the diameter coefficient is low. On the other hand, if the diameter coefficient is high (Fig. 11c, and d), some letters of the alphabet are omitted incorrectly. Therefore, choosing the suitable diameter coefficient is very important (Fig. 11b).



دیکانستراکشن که یکی از زیر مجموعه های مکتب پساساختارگرایی محسوب می
شود، نقدی به بینش ساختارگرایی و همچنین تفکر مدرن است .این مکتب فکری
توسط ژاک دریدا فیلسوف معاصر فرانسوی پایه گذاری شد .دریدا با ساختارها
مخالف اســت و اعتقاد دارد که وقتی ما به دنبال ســاختارها هســتیم از متغیرها
غافل می مانیم .فرهنگ و شــیوه های قومی هر لحظه تغییر می کند، پس روش
ســاختار گراها نمی تواند صــحیح باشــد .به عقیده دریدا یک متن هرگز مفهوم
واقعی خودش را آشکار نمی کند، زیرا مولف آن متن حضور ندارد و هر خواننده و
یا هر کس که آن متن را قرائت می کند، می تواند دریافتی متفاوت از قصــد و
هدف مولف داشته باشد

Fig. 10 Sample of first dataset image

Table 3 shows in more detail than Fig. 11 the error rate of each coefficient in separating points and main text. As shown in Table 3, the value 10 is the best possible coefficient among coefficients and has the lowest error rate among coefficients. Besides, we calculated the confusion matrix for the coefficient with the value of 10. Table 4 shows the confusion matrix for 50 images of the first dataset, which contains nearly 50,000 labels. The number of correct and incorrect predictions are summarized with count values and broken down by



(a) Using 5 as the coefficient value.

(b) Using 10 as the coefficient value.

(c) Using 15 as the coefficient value.

(d) Using 20 as the coefficient value.

Fig. 11 Removing points and small objects from image

Table 3 The error rate of each coefficient in separating points and main text

Coefficient value	Point error	Main text error	Total error	error rate
5	2790	70	2860	0.05
10	960	196	1156	**0.03**
15	840	508	1348	0.07
20	674	1862	2536	0.05

Highest accuracy or lowest error unique is in bold

each class. In addition, the ROC curve of these four coefficients is shown in Fig. 12. The recall and precision calculation are as follows:

$$Precision_i = \frac{M_{ii}}{\sum_i M_{ij}} \tag{5}$$

$$Recall_i = \frac{M_{ii}}{\sum_j M_{ij}} \tag{6}$$

One of the challenges we have been facing in this step was finding the best searching length, especially when the image has more than one column containing text (Fig. 13a). As shown in Fig. 13c, because of the short searching length, sometimes the searching method cannot find any CCs around the main CC to connect them. As a result, the searching algorithm cannot detect the line correctly. As shown in Fig. 13b, the opposite can also happen, especially if the image has multiple columns.

As we said in the last paragraph, by increasing or decreasing the searching length, the method encountered different kinds of problems. These problems are exacerbated in curve lines (Fig. 14a). Short searching length poses the same problem as previously described and could not recognize all the CCs of a line correctly (Fig. 14c). But, as shown in Fig. 14b, by increasing the searching length, the method could not separate two consecutive lines in a column correctly and connects them. Using this adaptive coefficient and multiplying it by the final font size, the method does not need to do anything more to detect curve lines (Fig. 14d).

4.3 Experimental result

Table 5 shows the performance results of our method versus five widely used methods for text-line extractions: i)Smearing [23], ii)XY cut [45], iii)Voronoi diagram-based [37], iv)Seam curving [52], vi)Docstrum [47]. In this part, we obtain these results when tested on our dataset with six different font types and Bold styles. As shown in Table 5, our

Table 4 Confusion matrix for the coefficient with the value of 10

	Predicted: Main text	Predicted: small obj	Recall
Actual: Main text	29966	196	99.35%
Actual: small obj	960	18370	95.03%
Precision	96.89%	98.94%	

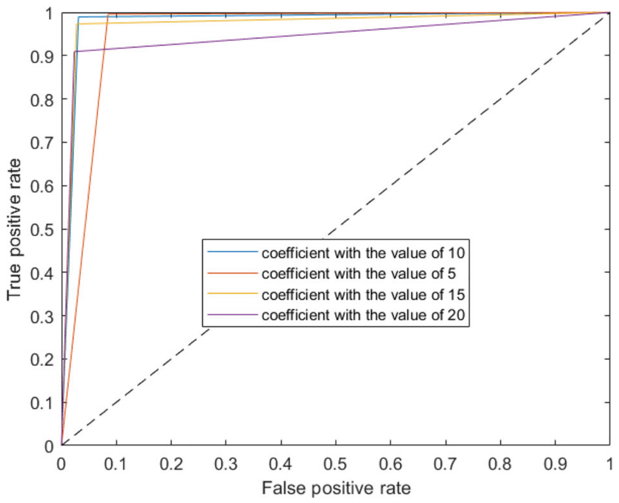
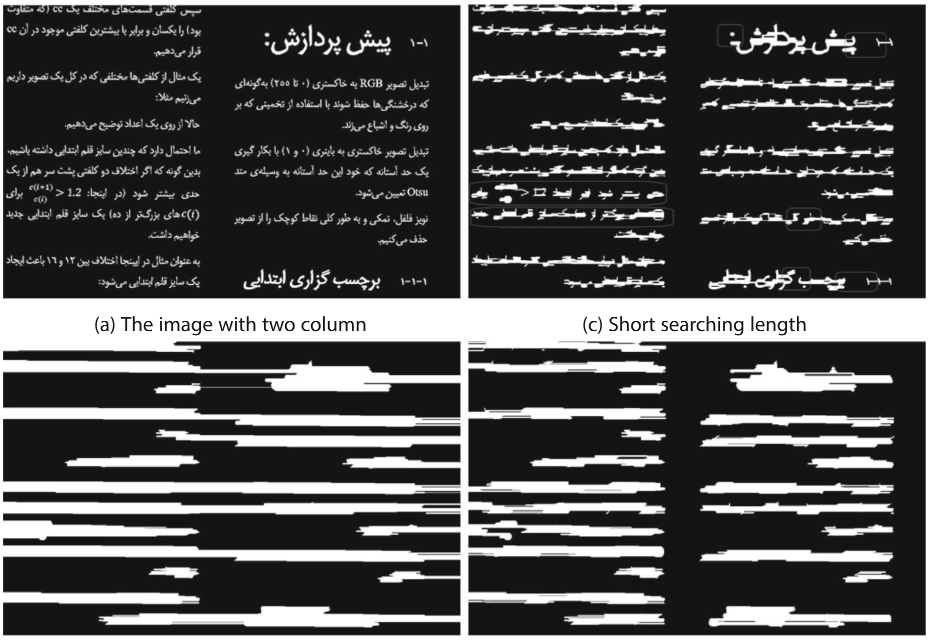


Fig. 12 ROC curves for coefficient with the value of 5, 10, 15, and 20

method has the best performance in all cases. We have also used formula (7) to calculate the accuracy in each case:

$$accuracy_{i,j} = \frac{\#oflinescorrectlysegmented}{totlal\#oflines},\ i \in font\ styles,\ j \in font\ types \quad (7)$$



۱-۱ پیش پردازش:

تبدیل تصویر RGB به خاکستری (۰ تا ۲۵۵) به‌گونه‌ای که درخشندگی‌ها حفظ شوند با استفاده از تخمینی که بر روی رنگ و اشباع می‌زند.

تبدیل تصویر خاکستری به باینری (۰ و ۱) با بکار گیری یک حد آستانه که خود این حد آستانه به وسیله‌ی متد Otsu تعیین می‌شود.

نویز فلفل. نمکی و به طور کلی نقاط کوچک را از تصویر حذف می‌کنیم.

۱-۱-۱ برچسب گزاری ابتدایی

سپس کلفتی قسمت‌های مختلف یک cc (که متفاوت بود) را یکسان و برابر با بیشترین کلفتی موجود در آن cc قرار می‌دهیم.

یک مثال از کلفتی‌ها مختلفی که در کل یک تصویر داریم می‌زنیم مثلا:

حالا از روی یک اعداد توضیح می‌دهیم.

ما احتمال دارد که چندین سایز قلم ابتدایی داشته باشیم، بدین گونه که اگر اختلاف دو کلفتی پشت سر هم از یک حدی بیشتر شود (در اینجا: $\frac{c(i+1)}{c(i)} > 1.2$ برای $c(i)$های بزرگتر از ده) یک سایز قلم ابتدایی جدید خواهیم داشت.

به عنوان مثال در اینجا اختلاف بین ۱۲ و ۱۶ باعث ایجاد یک سایز قلم ابتدایی می‌شود:

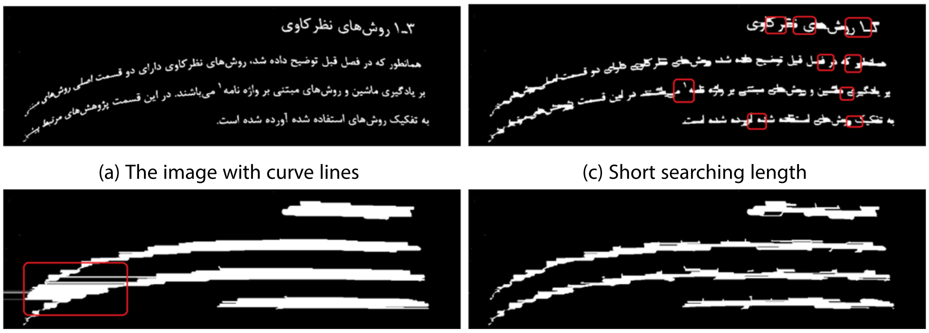
(a) The image with two column

(c) Short searching length

(b) Long searching length

(d) Currect searching length

Fig. 13 Searching length problem



(a) The image with curve lines

(c) Short searching length

(b) Long searching length

(d) Currect searching length

Fig. 14 Searching length problem

Table 5 The Bold style accuracy of proposed method and five states of the art methods for different font types

Font type	Smearing	X-Y Cut	Voronoi	Seam curving	Docstrum	Proposed method
Arial	99.23%	95%	95.99%	30.61%	69.74	**99.51%**
Nazanin	98.89%	95%	95.98%	30.05%	71.06	**99.44%**
Ziba	98.94%	96.02%	96.99%	32.58%	64.83	**99.35%**
IranNastaliq	98.87%	95.84%	95.59%	29.26%	57.76	**99.24%**
Tahoma	99.04%	98.02%	98.02%	32.14%	54.99	**99.41%**
XB-Niloofar	98.84%	96.99%	96.03%	28.22%	62.81	**99.45%**
Total accuracy	98.98%	96.28%	96.57%	30.6%	63.49%	**99.42%**

Highest accuracy or lowest error unique are in bold

Table 6 The Italic style accuracy of proposed method and five states of the art methods for different font types

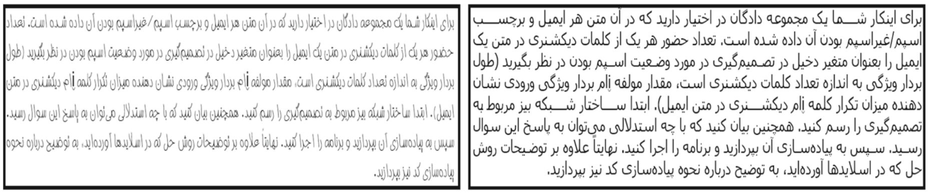
Font type	Smearing	X-Y Cut	Voronoi	Seam curving	Docstrum	Proposed method
Arial	99.07%	96.01%	95.01%	28.67%	66.69	**99.71%**
Nazanin	98.69%	97.03%	95.03%	29.95%	67.7	**99.1%**
Ziba	**99.59%**	96.98%	96.01%	31.05%	61.78	98.94%
IranNastaliq	98.1%	95.07%	94.82%	28.95%	53.73	**98.36%**
Tahoma	98.75%	96.14%	95.01%	33.71%	57.74	**99.47%**
XB-Niloofar	98.88%	97%	97.98%	28.1%	59.62	**99.3%**
Total accuracy	98.89%	96.46%	95.68%	30.26%	61.68%	**99.22%**

Highest accuracy or lowest error unique are in bold

Table 7 The Regular style accuracy of proposed method and five states of the art methods for different font types

Font type	Smearing	X-Y Cut	Voronoi	Seam curving	Docstrum	Proposed method
Arial	99.14%	98%	97%	28.74%	66.98	**99.71%**
Nazaninzanin	99.03%	97.03%	95.03%	29.81%	67.98	**99.38%**
Ziba	**99.67%**	96.01%	97.96%	30.81%	62.02	98.94%
IranNastaliq	98.23%	94.94%	94.56%	29.08%	54.11	**98.36%**
Tahoma	98.81%	96.97%	97.98%	33.47%	57.98	**99.41%**
XB-Nilofar	99.02%	96.03%	94.98%	32.64%	59.97	**99.3%**
Total accuracy	99.02%	96.64%	96.39%	30.98%	61.97%	**99.26%**

Highest accuracy or lowest error unique are in bold


برای اینکار شما یک مجموعه دادگان در اختیار دارید که در آن متن هر ایمیل و برچسب اسپم/غیراسپم بودن آن داده شده است. تعداد
حضور هر یک از کلمات دیکشنری در متن یک ایمیل را بعنوان متغیر دخیل در تصمیم‌گیری در مورد وضعیت اسپم بودن در نظر بگیرید (طول
بردار ویژگی به اندازه تعداد کلمات دیکشنری است، مقدار مولفه iام بردار ویژگی ورودی نشان دهنده میزان تکرار کلمه iام دیکشنری در متن
ایمیل). ابتدا ساختار شبکه بیز مربوط به تصمیم‌گیری را رسم کنید. همچنین بیان کنید که با چه استدلالی می‌توان به پاسخ این سوال رسید.
سپس به پیاده‌سازی آن بپردازید و برنامه را اجرا کنید. نهایتاً علاوه بر توضیحات روش حل که در اسلایدها آورده‌اید، به توضیح درباره نحوه
پیاده‌سازی کد نیز بپردازید.

(a) Ziba font with size 14

برای اینکار شـــما یک مجموعه دادگان در اختیار دارید که در آن متن هر ایمیل و برچســـب
اسپم/غیراسپم بودن آن داده شده است. تعداد حضور هر یک از کلمات دیکشنری در متن یک
ایمیل را بعنوان متغیر دخیل در تصـمیم‌گیری در مورد وضـعیت اسـپم بودن در نظر بگیرید (طول
بردار ویژگی به اندازه تعداد کلمات دیکشنری است، مقدار مولفه iام بردار ویژگی ورودی نشان
دهنده میزان تکرار کلمه iام دیکشـــنری در متن ایمیل). ابتدا ســـاختار شـــبکه بیز مربوط به
تصمیم‌گیری را رسم کنید. همچنین بیان کنید که با چه استدلالی می‌توان به پاسخ این سوال
رسـید. سـپس به پیاده‌سـازی آن بپردازید و برنامه را اجرا کنید. نهایتاً علاوه بر توضـیحات روش
حل که در اسلایدها آورده‌اید، به توضیح درباره نحوه پیاده‌سازی کد نیز بپردازید.

(b) Tahoma font with size 14

Fig. 15 Ziba font problem

Table 8 The Bold style accuracy of proposed method and five states of the art methods for different font sizes

Font size	Smearing	X-Y Cut	Voronoi	Seam curving	Docstrum	Proposed method
8	95.81%	93.48%	93.48%	21.34%	54.99%	**97.3%**
10	97.43%	94.85%	95.25%	26.87%	57.68%	**99.03%**
14	**99.75%**	96.55%	96.86%	28.56%	60.83%	99.69%
18	99.9%	96.6%	97.05%	33.42%	65.72%	**99.91%**
22	99.96%	97.86%	98.08%	35.84%	70.4%	**100%**
Total accuracy	98.98%	96.28%	96.57%	30.6%	63.49%	**99.42%**

Highest accuracy or lowest error unique are in bold

Table 9 The Italic style accuracy of proposed method and five states of the art methods for different font sizes

Font size	Smearing	X-Y Cut	Voronoi	Seam curving	Docstrum	Proposed method
8	95.61%	93.42%	93.8%	19.56%	51.24%	**96.37%**
10	97.94%	95.47%	93.99%	25.68%	54.9%	**98.68%**
14	99.29%	95.94%	94.9%	28.45%	59.03%	**99.61%**
18	99.79%	96.97%	96.39%	33.49%	64.63%	**99.95%**
22	99.87%	98.32%	97.39%	36.18%	69.48%	**99.96%**
Total accuracy	98.89%	96.46%	95.68%	30.26%	61.68%	**99.22%**

Highest accuracy or lowest error unique are in bold

Table 10 The Regular style accuracy of proposed method and five states of the art methods for different font sizes

Font size	Smearing	X-Y Cut	Voronoi	Seam curving	Docstrum	Proposed method
8	96.18%	93.7%	94.56%	21.09%	51.53%	**96.37%**
10	98.02%	95.72%	94.9%	26.5%	55.14%	**98.77%**
14	99.42%	96.13%	95.74%	29.03%	59.35%	**99.68%**
18	99.84%	97.07%	96.87%	34.01%	64.94%	**99.95%**
22	99.91%	98.5%	98.1%	36.75%	69.75%	**100%**
Total accuracy	99.02%	96.64%	96.39%	30.98%	61.97%	**99.26%**

Highest accuracy or lowest error unique are in bold

Table 11 The text lines recognition accuracy with an average runtime of the proposed method and other methods on the Arabic OCR dataset

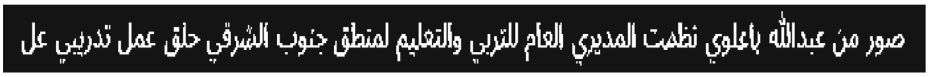
Methods	Accuracy	Average runtime per image (s)
Kraken	98.75%	23.22
OCRopus	99.1%	18.24
Smearing	95.1%	14.45
Proposed method	**99.24%**	**8.33**

Highest accuracy or lowest error unique are in bold

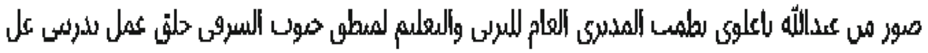
Table 12 The text lines recognition accuracy with an average runtime of the proposed method and other methods on dataset 3

Methods	Accuracy	Average runtime per image (s)
Kraken	91.23%	28.43
OCRopus	94.51%	25.33
Smearing	85.88%	19.44
Proposed method	**98.35%**	**15.55**

Highest accuracy or lowest error unique are in bold

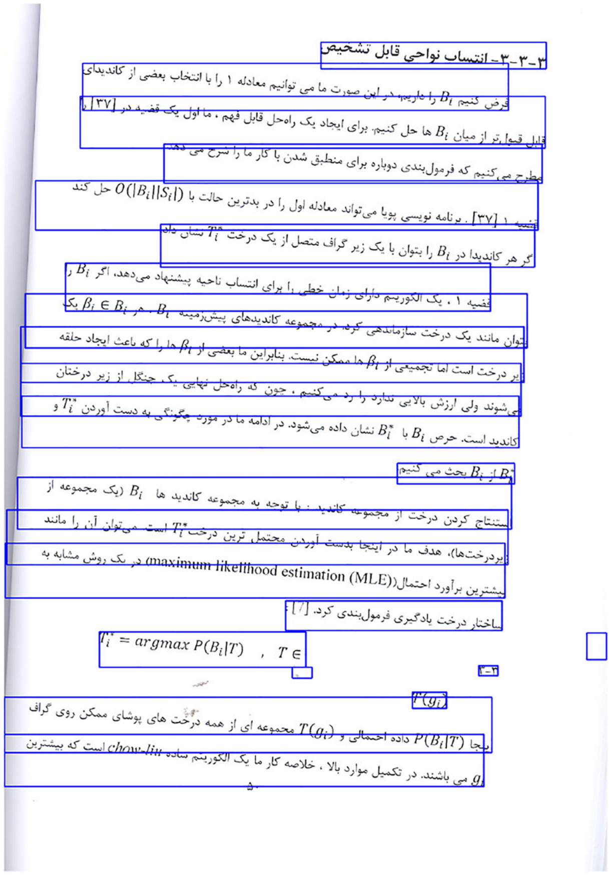

صور من عبدالله باعلوي نظمت المديري العام للتربي والتعليم لمنطق جنوب الشرقي حلق عمل تدريبي عل

(a) output line in the proposed method


صور من عبدالله باعلوى بطمب المدبرى العام للبرى والبعلم لمطق حبوب السرقى حلق عمل بدربى عل

(b) output line in the OCRopus method

Fig. 16 Problem of removing small objects in OCRopus method



۳-۳-۳- انتساب نواحی قابل تشخیص

فرض کنیم B_i را داریم، در این صورت ما می توانیم معادله ۱ را با انتخاب بعضی از کاندیدای
قابل قبول‌تر از میان B_i ها حل کنیم. برای ایجاد یک راه‌حل قابل فهم ، ما اول یک قضیه در [۳۷] را
مطرح می‌کنیم که فرمول‌بندی دوباره برای منطبق شدن با کار ما را شرح می دهد.
قضیه ۱ [۳۷] . برنامه نویسی پویا می‌تواند معادله اول را در بدترین حالت با $O(|B_i||S_i|)$ حل کند
اگر هر کاندیدا در B_i را بتوان با یک زیر گراف متصل از یک درخت T_i^* نشان داد.
قضیه ۱ . یک الگوریتم دارای زمان خطی را برای انتساب ناحیه پیشنهاد می‌دهد، اگر B_i را
بتوان مانند یک درخت سازماندهی کرد. در مجموعه کاندیدهای پیش‌زمینه B_i ، هر $\beta_i \in B_i$ یک
زیر درخت است اما تجمیعی از β_i ها ممکن نیست. بنابراین ما بعضی از β_i ها را که باعث ایجاد حلقه
می‌شوند ولی ارزش بالایی ندارد را رد می‌کنیم ، چون که راه‌حل نهایی یک جنگل از زیر درختان
کاندید است. حرص B_i با B_i^* نشان داده می‌شود. در ادامه ما در مورد چگونگی به دست آوردن T_i^* و
B_i^* از B_i بحث می کنیم

استنتاج کردن درخت از مجموعه کاندید : با توجه به مجموعه کاندید ها B_i (یک مجموعه از
زیردرخت‌ها)، هدف ما در اینجا بدست آوردن محتمل ترین درخت T_i^* است. می‌توان آن را مانند
بیشترین برآورد احتمال((maximum likelihood estimation (MLE) در یک روش مشابه به
ساختار درخت یادگیری فرمول‌بندی کرد. [۷]؛

$$T_i^* = argmax\, P(B_i|T) \quad , \quad T \in$$

۳-۲

اینجا $P(B_i|T)$ داده احتمالی و $T(g_i)$ مجموعه ای از همه درخت های پوشای ممکن روی گراف
g_i می باشند. در تکمیل موارد بالا ، خلاصه کار ما یک الگوریتم ساده $chow\text{-}liu$ است که بیشترین

۵۰

Fig. 17 The output of the Kraken method on dataset 3

Also, Tables 6 and 7 show the performance results of our method versus five widely used methods for text-line extractions. As shown in Tables 6 and 7, our method has the best performance in all cases except the "Ziba" font. As shown in Fig. 15a and b, even though the "Ziba" and "Tahoma" font sizes are the same, the diameter of the "Ziba" font is smaller than the "Tahoma" font. As a result, the final-font-size of the "Ziba" font, and the searching-length, are smaller. As shown in these tables, as much the font size is larger, the easier it is to recognize text lines, as the number of words within a line decreases and the final font size increases.

Tables 8, 9, and 10 show the performance results of our method versus five widely used methods for text-line extractions: i)Smearing [23], ii)XY cut [45], iii)Voronoi diagram-based [37], iv)Seam curving [52], vi)Docstrum [47]. In this part, we obtain these results when tested on our dataset with five different font sizes and three styles: Bold, Regular, and Italic. As shown in Tables 8, 9 and 10, our method has the best performance in most cases.

As mentioned beforehand, we used two more datasets to evaluate the performance of the proposed method further. In addition to the proposed method, we test the smearing method, which is the best method among the related works, and the OCRopus and the Kraken methods, which are more up-to-date than other methods, on these two datasets.

Tables 11 and 12 show the accuracy with average runtime of the proposed method and other methods on the Arabic OCR dataset and the dataset 3. As shown in Tables 11 and 12, the proposed method has the best performance among all methods.

Due to the high complexity of the third dataset compared to the other two datasets, the accuracy of the methods in this dataset is lower. As shown in formula (7), the accuracy of each method represents the percentage of the number of lines that have been extracted correctly. The OCRopus method, despite its good accuracy in detecting lines, also has many disadvantages.

One of the main problems with the OCRopus method is its high runtime, which can take up to 30 seconds in images with many CCs. This high runtime is one of the main problems of this method, but the problems do not end here. One of the most critical problems of this method is removing small objects and points in the image. Figure 16a is one of the output lines in the proposed method. Figure 16b shows the output of the same line in the OCRopus method. As can be seen, most of the points were removed in this method. Failure to run the program in images with high resolution (more than 300 dpi) is another problem of this method. The proposed method is specific to the Persian language. In Persian, the number of dots is much higher than in languages such as English. Therefore, in the proposed method, the dots are well kept.

Same as the OCRopus method, the Kraken method has high runtime. The Kraken method has low accuracy on images that are skewed or curved. For this reason, it does not work well on dataset 3. Figure 17 shows an example of the output of the Kraken method on dataset3. As shown in Fig. 17, most lines overlap, which reduces the performance of the OCR system.

Both OCRopus and Kraken methods have difficulty detecting images that are skewed or curved. While the skewness of the lines or their curvature has less effect on our method. Our method is resistant to curvature and skew due to the use of the final font size.

5 Conclusion

In this paper, we introduced a new text-line detection method for printed text images by using the final font size. The proposed method measures the diameters of each CC from any direction, then chooses the maximum of these diameters as the font size of that CC. The algorithm tries to select one of the font sizes as the final font size by defining an adaptive parameter. Finally, the algorithm tries to connect any CCs that are horizontally in the same direction. The proposed method is tested on more than 24000 lines with different font types, sizes, and styles. The proposed method can process document images with the following points: i) Persian text with and without diacritics; ii) non-linear warping text lines; iii) skew text lines; iv) six Persian font types: Nazanin, Ziba, IranNastaliq, XB-Niloofar, Arial, and Tahoma; v) Bold, italic and regular style; vi) and different font sizes 8,10,14,18,22 including the mix between them. As the method has good results and reached 99.3% accuracy on our dataset, we still need to do more evaluation on other Persian fonts and extend this to Arabic and other languages. Also, text line segmentation for handwriting documents and scene text detection are some other challenges of our future purpose.

References

1. Ahmad I, Wang X, Li R, Ahmed M, Ullah R (2017) Line and ligature segmentation of urdu nastaleeq text. IEEE Access 5:10924–10940

2. Alaei A, Nagabhushan P, Pal U (2011) Piece-wise painting technique for line segmentation of unconstrained handwritten text: a specific study with persian text documents. Pattern Anal Appl 14(4):381–394
3. Aljarrah I, Al-Khaleel O, Mhaidat K, Alrefai M, Alzu'bi A, Rabab'ah M (2012) Automated system for arabic optical character recognition. In: Proceedings of the 3rd International Conference on Information and Communication Systems. pp 1–6
4. Ayesh M, Mohammad K, Qaroush A, Agaian S, Washha M (2017) A robust line segmentation algorithm for arabic printed text with diacritics. Electron Imaging 2017(13):42–47
5. Banumathi K, Chandra AJ (2016) Line and word segmentation of kannada handwritten text documents using projection profile technique. In: 2016 International Conference on Electrical, Electronics, Communication, Computer and Optimization Techniques (ICEECCOT). IEEE, pp 196–201
6. Breuel TM (2008) The ocropus open source ocr system. In: Document recognition and retrieval XV, vol 6815. International Society for Optics and Photonics, p 68150F
7. Brodić D, Milivojević ZN (2013) Text line segmentation with the algorithm based on the oriented anisotropic gaussian kernel. J Electr Eng 64(4):238–243
8. Brown MS, Seales WB (2004) Image restoration of arbitrarily warped documents. IEEE Trans Pattern Anal Mach Intell 26(10):1295–1306
9. Bukhari SS, Shafait F, Breuel TM (2013) Coupled snakelets for curled text-line segmentation from warped document images. Int J Doc Anal Recog (IJDAR) 16(1):33–53
10. Bukhari SS, Shafait F, Breuel TM (2009) Adaptive binarization of unconstrained hand-held camera-captured document images. J UCS 15(18):3343–3363
11. Bukhari SS, Shafait F, Breuel TM (2009) Ridges based curled textline region detection from grayscale camera-captured document images. In: International Conference on Computer Analysis of Images and Patterns. Springer, pp 173–180
12. Bukhari SS, Shafait F, Breuel TM (2009) Coupled snakelet model for curled textline segmentation of camera-captured document images. In: 2009 10th International Conference on Document Analysis and Recognition. IEEE, pp 61–65
13. Bustacara-Medina C, Florez-Valencia L, Diaz LC (2020) Improved canny edge detector using principal curvatures. J Electr Electron Eng 8(4):109
14. Cheng Q, Wang G, Dong Q, Wei B (2020) Arbitrary-shaped text detection with adaptive convolution and path enhancement pyramid network. Multimedia Tools Appl 79(39):29225–29242
15. Chernyshova YS, Sheshkus AV, Arlazarov VV (2020) Two-step cnn framework for text line recognition in camera-captured images. IEEE Access 8:32587–32600
16. Cheung A, Bennamoun M, Bergmann NW (2001) An arabic optical character recognition system using recognition-based segmentation. Pattern Recog 34(2):215–233
17. Diem M, Kleber F, Sablatnig R (2013) Text line detection for heterogeneous documents. In: 2013 12th International Conference on Document Analysis and Recognition. IEEE, pp 743–747
18. El Bahi H, Zatni A (2019) Text recognition in document images obtained by a smartphone based on deep convolutional and recurrent neural network. Multimedia Tools Appl 78(18):26453–26481
19. Ezaki H, Uchida S, Asano A, Sakoe H (2005) Dewarping of document image by global optimization. In: Eighth International Conference on Document Analysis and Recognition (ICDAR'05). IEEE, pp 302–306
20. Fateh A (2021) Persian dataset of scanned images. https://drive.google.com/file/d/1czMAGodDxBDQajNfSdYibFBJ9pooa69_/view?usp=sharing
21. Fateh A (2021) Persian dataset in different font types, sizes, and styles. https://drive.google.com/file/d/1jaDp7qI6480yNImRZQpkY_aOJ8o7mv8J/view?usp=sharing
22. Fakhari A, Kiani K (2021) A new restricted boltzmann machine training algorithm for image restoration. Multimed Tools Appl 80(2):2047–2062
23. Forczmański P, Markiewicz A (2016) Two-stage approach to extracting visual objects from paper documents. Mach Vis Appl 27(8):1243–1257
24. Garg B (2020) Restoration of highly salt-and-pepper-noise-corrupted images using novel adaptive trimmed median filter. SIViP 14:1555–1563
25. Garg R, Garg NK (2014) A new approach for line segmentation in punjabi language using strip based projection profile method
26. Gatos B, Pratikakis I, Ntirogiannis K (2007) Segmentation based recovery of arbitrarily warped document images. In: Ninth International Conference on Document Analysis and Recognition (ICDAR 2007), vol 2. IEEE, pp 989–993
27. Grana C, Serra G, Manfredi M, Coppi D, Cucchiara R (2016) Layout analysis and content enrichment of digitized books. Multimed Tools Appl 75(7):3879–3900

28. Grüning T, Leifert G, Strauß T, Michael J, Labahn R (2019) A two-stage method for text line detection in historical documents. Int J Doc Anal Recog (IJDAR) 22(3):285–302
29. Guo D, Qu X, Du X, Wu K, Chen X (2014) Salt and pepper noise removal with noise detection and a patch-based sparse representation. Adv Multimed 2014:
30. Guo Y, Sun Y, Bauer P, Allebach JP, Bouman CA (2015) Text line detection based on cost optimized local text line direction estimation. In: Color Imaging XX: Displaying, Processing, Hardcopy, and Applications, vol. 9395. International Society for Optics and Photonics, p 939507
31. Gupta N, Jalal AS (2019) A robust model for salient text detection in natural scene images using mser feature detector and grabcut. Multimedia Tools Appl 78(8):10821–10835
32. Hall M, Frank E, Holmes G, Pfahringer B, Reutemann P, Witten IH (2009) The weka data mining software: an update. ACM SIGKDD Explor Newsl 11(1):10–18
33. Hussain S, Ali S et al (2015) Nastalique segmentation-based approach for urdu ocr. Int J Doc Anal Recog (IJDAR) 18(4):357–374
34. Jo J, Koo HI, Soh JW, Cho NI (2020) Handwritten text segmentation via end-to-end learning of convolutional neural networks. Multimed Tools Appl 79(43):32137–32150
35. Kaur RP, Kumar M, Jindal MK (2019) Newspaper text recognition of gurumukhi script using random forest classifier. Multimedia Tools Appl 1–14
36. Kchaou MG, Kanoun S, Ogier J-M (2012) Segmentation and word spotting methods for printed and handwritten arabic texts: a comparative study. In: 2012 international conference on frontiers in handwriting recognition. IEEE, 274–279
37. Kise K, Sato A, Iwata M (1998) Segmentation of page images using the area voronoi diagram. Comput Vis Image Underst 70(3):370–382
38. Koo HI (2016) Text-line detection in camera-captured document images using the state estimation of connected components. IEEE Trans Image Process 25(11):5358–5368
39. Last release of ocropus (2017). https://github.com/ocropus/ocropy
40. Last release of kraken (2021). https://github.com/mittagessen/kraken
41. Lavialle O, Molines X, Angella F, Baylou P (2001) Active contours network to straighten distorted text lines. In: Proceedings 2001 International Conference on Image Processing (Cat. No. 01CH37205), vol. 3. IEEE, pp 748–751
42. Lyu B, Akama R, Tomiyama H, Meng L (2019) The early japanese books text line segmentation base on image processing and deep learning. In: 2019 International Conference on Advanced Mechatronic Systems (ICAMechS). IEEE, pp 299–304
43. Mahmood A, Srivastava A (2018) A novel segmentation technique for urdu type-written text. In: 2018 Recent Advances on Engineering, Technology and Computational Sciences (RAETCS). IEEE, pp 1–5
44. Malakar S, Halder S, Sarkar R, Das N, Basu S, Nasipuri M (2012) Text line extraction from handwritten document pages using spiral run length smearing algorithm. In: 2012 International Conference on Communications, Devices and Intelligent Systems (CODIS). IEEE, pp 616–619
45. Nagy G, Seth S, Viswanathan M (1992) A prototype document image analysis system for technical journals. Computer 25(7):10–22
46. Nguyen TT, Dai Pham X, Kim D, Jeon JW (2008) A test framework for the accuracy of line detection by hough transforms. In: 2008 6th IEEE international conference on industrial informatics. IEEE, pp 1528–1533
47. O'Gorman L (1993) The document spectrum for page layout analysis. IEEE Trans Pattern Anal Mach Intell 15(11):1162–1173
48. Otsu N (1979) A threshold selection method from gray-level histograms. IEEE transactions on systems, man, and cybernetics 9(1):62–66
49. Paul S, Saha S, Basu S, Saha PK, Nasipuri M (2019) Text localization in camera captured images using fuzzy distance transform based adaptive stroke filter. Multimedia Tools Appl 78(13):18017–18036
50. Rahmati M, Fateh M, Rezvani M, Tajary A, Abolghasemi V (2020) Printed persian ocr system using deep learning. IET Image Processing 14(15):3920–3931
51. Rais M, Goussies NA, Mejail M (2011) Using adaptive run length smoothing algorithm for accurate text localization in images. In: Iberoamerican Congress on Pattern Recognition. Springer, pp 149–156
52. Seuret M, Ben Ezra DS, Liwicki M (2017) Robust heartbeat-based line segmentation methods for regular texts and paratextual elements. In: Proceedings of the 4th international workshop on historical document imaging and processing, pp 71–76
53. Shahab A, Shafait F, Dengel A (2011) Icdar 2011 robust reading competition challenge 2: Reading text in scene images. In: 2011 international conference on document analysis and recognition, IEEE, pp 1491–1496
54. Shaikh NA, Mallah GA, Shaikh ZA (2009) Character segmentation of sindhi, an arabic style scripting language, using height profile vector. Aust J Basic Appl Sci 3(4):4160–4169

55. Soni R, Kumar B, Chand S (2019) Optimal feature and classifier selection for text region classification in natural scene images using weka tool. Multimedia Tools Appl 78(22):31757–31791
56. Soujanya P, Koppula VK, Gaddam K, Sruthi P (2010). In: Comparative study of text line segmentation algorithms on low quality documents. CMR College of Engineering and Technology Cognizant Technologies, Hyderabad, India
57. Tan CL, Zhang L, Zhang Z, Xia T (2005) Restoring warped document images through 3d shape modeling. IEEE Trans Pattern Anal Mach Intell 28(2):195–208
58. Ulges A, Lampert CH, Breuel TM (2005) Document image dewarping using robust estimation of curled text lines. In: Eighth International Conference on Document Analysis and Recognition (ICDAR'05). IEEE, pp 1001–1005
59. Wang X, Song Y, Zhang Y, Xin J (2017) A hierarchical recursive method for text detection in natural scene images. Multimedia Tools Appl 76(24):26201–26223
60. Wu C, Agam G (2002) Document image de-warping for text/graphics recognition. In: Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR). Springer, pp 348–357
61. Yin X-C, Zuo Z-Y, Tian S, Liu C-L (2016) Text detection, tracking and recognition in video: a comprehensive survey. IEEE Trans Image Process 25(6):2752–2773
62. Youssef H (2020) Arabic dataset ocr. https://drive.google.com/drive/folders/1–wsm4NIZB8Reu70jg-wBO56Pq%89N6fs
63. Zeki AM, Zakaria MS, Liong C-Y (2013) Segmentation of arabic characters: A comprehensive survey. In: Technology Diffusion and Adoption: Global Complexity, Global Innovation. IGI Global, pp 251–288
64. Zhang Z, Tan CL (2003) Correcting document image warping based on regression of curved text lines. In: Seventh International Conference on Document Analysis and Recognition, 2003. Proceedings. IEEE, pp 589–593