Chapter 2
Layout Analysis of Arabic Script Documents

Syed Saqib Bukhari, Faisal Shafait, and Thomas M. Breuel

Abstract Layout analysis—extraction of text lines from a document image and identification of their reading order—is an important step in converting the document into a searchable electronic representation. Projection methods are typically employed for extraction of text lines in Arabic script documents. Although projection methods achieve good accuracy on clean, skew-free documents, their performance drops under challenging situations (border noise, skew, complex layouts, etc.). This chapter presents a layout analysis system for extracting text lines in reading order from scanned Arabic script document images written in different languages (Arabic, Urdu, Persian, etc.) and different styles (Naskh, Nastaliq, etc.). The presented system is based on a suitable combination of different well-established techniques for analyzing Latin script documents that have proven to be robust against different types of document image degradations.

2.1 Introduction

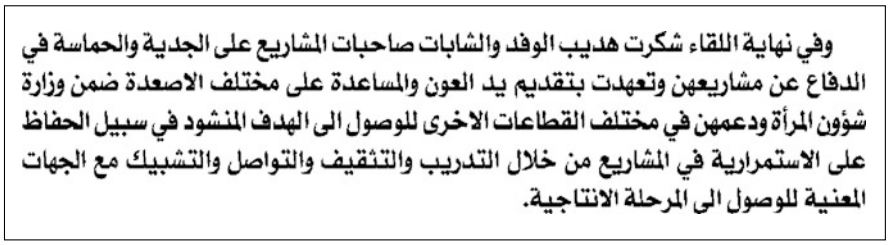
Layout analysis deals with text line detection and their reading order determination in document images. The wide variety of layouts in large-scale document digitization projects poses stern challenges to document image analysis. A document image may contain different types of contents like text, graphics, halftones, etc. The goal of optical character recognition (OCR) is to extract text from a document image. This is achieved in two steps. The first step, geometric layout analysis, locates text lines in the image and identifies their reading order. In the second step, text lines

S.S. Bukhari (✉) · T.M. Breuel
Technical University of Kaiserslautern, Kaiserslautern, Germany
e-mail: bukhari@informatik.uni-kl.de

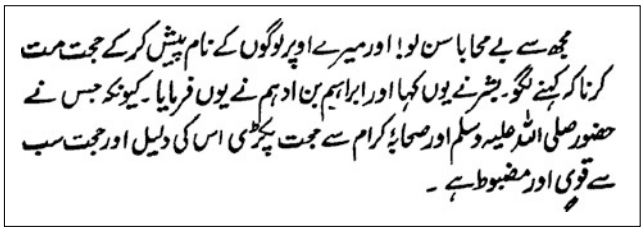
T.M. Breuel
e-mail: tmb@informatik.uni-kl.de

F. Shafait
German Research Center for Artificial Intelligence (DFKI), Kaiserslautern, Germany
e-mail: faisal.shafait@dfki.de

V. Märgner, H. El Abed (eds.), *Guide to OCR for Arabic Scripts*,
DOI 10.1007/978-1-4471-4072-6_2,


وفي نهاية اللقاء شكرت هديب الوفد والشابات صاحبات المشاريع على الجدية والحماسة في الدفاع عن مشاريعهن وتعهدت بتقديم يد العون والمساعدة على مختلف الاصعدة ضمن وزارة شؤون المرأة ودعمهن في مختلف القطاعات الاخرى للوصول الى الهدف المنشود في سبيل الحفاظ على الاستمرارية في المشاريع من خلال التدريب والتثقيف والتواصل والتشبيك مع الجهات المعنية للوصول الى المرحلة الانتاجية.

(a) Sample Arabic document written in Naskh script.


مجھ سے بے محابا سن لو! اور میرے اوپر لوگوں کے نام پیش کر کے حجت مت کرنا کہ کہنے لگو۔ بشر نے یوں کہا اور ابراہیم بن ادہم نے یوں فرمایا۔ کیونکہ جس نے حضور صلی اللہ علیہ وسلم اور صحابۂ کرام سے حجت پکڑی اس کی دلیل اور حجت سب سے قوی اور مضبوط ہے۔

(b) Sample Urdu document written in Nastaliq script.

Fig. 2.1 An example of printed Arabic text in Naskh script and Urdu text in Nastaliq script. Text lines in Nastaliq have very little spacing between them compared to Naskh script

identified by the layout analysis step are fed to a character recognition engine which converts them into text in an appropriate format (ASCII, UTF-8, etc.).

The Arabic script is used for writing several languages of Asia and Africa, e.g., Arabic, Urdu, Persian, Pashto, Kurdi, and Jawi. After Latin script, it is the second most widely used script in the world. It is a cursive script; i.e., individual characters are usually combined to form ligatures. Although there are many styles for writing Arabic script, the most widely used styles are Naskh and Nastaliq. The Naskh writing style is dominant in Arabic and Pashto languages, whereas Nastaliq is the standard style adopted for writing Urdu and Persian. Examples of printed Arabic text written in Naskh script and Urdu text written in Nastaliq script are shown in Fig. 2.1. From a layout analysis point of view, the main differences of Nastaliq script as compared to Naskh script are: (i) very small interline and interword spacing and (ii) tall ascenders and descenders that overlap into adjacent text lines.

Research on Arabic script OCR has primarily been focused on word recognition [1], and very few approaches have been proposed for text line extraction from machine printed Arabic script document images. Since Arabic is generally written in Naskh script, text line segmentation using horizontal projections works quite well on machine printed documents due to the large interline spacing [22]. Segmentation of a page image into individual lines by horizontal projection is a primitive approach and works only on clean, single-column documents with large interline spacing. To handle multi-column documents, either the $x-y$ cut method [29] is used, or morphological operations are employed to get text blocks [38], which can then be further subdivided into individual text lines by horizontal projection. More sophisticated approaches for text line extraction have been presented in the domain of segment-

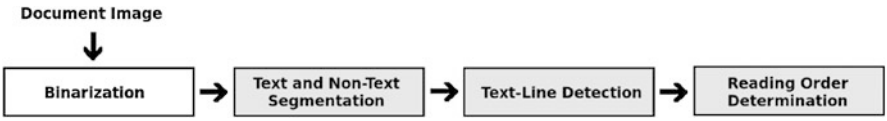


Fig. 2.2 Processing flow of a high performance generic layout analysis system. *Filled blocks* show the areas that this chapter discusses in detail

ing handwritten Arabic documents [4, 41]. However, the key problem addressed in these approaches is handling the local nonlinearity of text lines.

Over the last two decades, several layout analysis algorithms have been proposed in the literature (for a literature survey, please refer to [11, 28]) that work for different layouts and are quite robust to the presence of noise in the document. Many of these algorithms have come into widespread use for analyzing document images in different scripts. Kumar et al. [24] have evaluated the performance of six algorithms for page segmentation on Nastaliq script: the x–y cut [29], the smearing algorithm [40], whitespace analysis [2], the constrained text line finding algorithm [5], Docstrum [30], and the Voronoi-diagram based approach [23]. These algorithms work very well in segmenting documents in Latin script, as shown in [37]. However, when Kumar et al. applied these algorithms to segment Nastaliq script documents, none of these algorithms was able to achieve an accuracy of more than 70 % on their test data, which had simple book layouts with no font size variations within each page.

Contrary to Arabic OCR, there has been very little work in the area of Urdu or Persian document analysis. Husain et al. [19] proposed an Urdu character recognition system for the Nastaliq script. Urdu is written in Nastaliq script using more than 20,000 ligatures [16]. Husain et al. skipped the layout analysis step to concentrate more on the OCR part. Pal et al. [32] presented an approach for recognizing printed Urdu documents. First, they perform skew correction of the document using a Hough transform. Text lines in the skew corrected document are then segmented by horizontal projection. A similar approach is used by Jelodar et al. [20] to extract text lines from printed Persian documents.

Shafait et al. [36] have presented an adaptation of the layout system described in [6] to Urdu script documents. First, they evaluate empty whitespace rectangles as candidates for column separators or gutters. Text lines are then detected by modifying a RAST-based (Recognition by Adaptive Subdivision of Transformation Space) text-line finding algorithm [5] where column separators are introduced as "obstacles." Finally, text lines are analyzed for determining the reading order using constraints on the geometric arrangement of text line segments on the page. Particular advantages of their system are that it is nearly a parameter-free approach and robust to the presence of noise in document images.

In this chapter we present a layout analysis system that is an extension of the approach presented in [36] and is applicable to a wide variety of Arabic script binary document images. A grayscale document image can be first converted into binary form using an appropriate binarization approach such as those in Otsu [31] and Sauvola [35], which are commonly used state-of-the-art binarization approaches.

Fig. 2.3 A sample newspaper image printed in Arabic Naskh script and its corresponding layout analysis result: *gray region* represents non-text components, *color-coded labeling* represent segmented text lines, and *magenta line* shows text line reading order. [Note: *left* image has been taken from the website [17]]

A possible flow of a generic layout analysis system is shown in Fig. 2.2. Here, we will discuss text and non-text segmentation, text line detection, and reading order determination. For a sample Arabic script document image, the output of the layout analysis system is shown in Fig. 2.3.

The rest of the chapter is organized as follows. In Sect. 2.2, the multiresolution morphology-based text and non-text segmentation algorithm [3, 10] is described. State-of-the-art $x-y$ cut [29] and ridge-based [7] text line finding methods are explained in Sect. 2.3. A topological sorting-based reading order determination algorithm [36] is discussed in Sect. 2.4, followed by the conclusion in Sect. 2.5.

2.2 Text and Non-text Segmentation

Text and non-text segmentation is the process of separating text and non-text elements in document images. It is an important initial step in document image processing like optical character recognition (OCR) systems. A character recognition engine is designed for recognizing text elements, and it produces garbage for non-text elements.

Different approaches have been proposed in the literature for text and non-text segmentation. Wong et al. [40] presented a classical smearing-based page segmentation approach. Bloomberg [3] introduced a method for text and halftone segmentation using multiresolution morphology. Other state-of-the-art text and non-text segmentation approaches can be generally categorized as classification-based

approaches such as the pixel [27], connected component [9], or zone [21, 39] classification-based text and non-text segmentation methods. In general, the accuracy of classification-based text and non-text segmentation approaches heavily depends on the training samples.

The multiresolution morphology-based method [3] was specifically designed for separating halftones from document images. It works well for halftone segmentation, but not for other types of non-text elements like drawings, maps, etc. It is a simple approach, based on the assumption that the size of non-text elements is larger than text elements in document images. We presented an improvement to the multiresolution morphology-based text and non-text segmentation method in [10] which can also segment drawing type non-text elements. A data flow diagram of the improved version of Bloomberg's text and non-text segmentation method is shown in Fig. 2.4. A brief description of Bloomberg's multiresolution morphology-based text and non-text segmentation method [3] and our improved version [10] is described below.

Bloomberg introduced the concept of *threshold reduction* for subsampling of document images, which is defined as follows. Consider a binary document image where a foreground pixel is represented by 1 and a background pixel is represented by 0. Each 2×2 pixel block in the document image is replaced by a single value, either 1 or 0, in a corresponding subsampled image. The value is set to 1 if the sum of values in a particular 2×2 pixels block is greater than or equal to some predefined threshold value; otherwise the value is set to 0. This threshold reduction operation mimics the process of image dilation for a threshold value equal to 1 and erosion for a threshold value equal to 4 that follows subsampling of each 2×2 pixel block by its upper left pixel. The threshold reduction is also referred as multiresolution morphology.

Bloomberg used the concept of threshold reduction for implementing the text and halftone segmentation method that is shown in Fig. 2.4. An input image is first processed by two threshold reduction operations, both with threshold value equal to 1. These threshold reduction operations produce a subsampled image. The subsampled image is further processed by two threshold reductions with threshold values equal to 4 and 3, respectively, and a morphological opening operation. The output image is referred to as a seed image. The seed image, after expansion, is compared with the subsampled image for generating a halftone-mask image. The halftone-mask image is composed of fully or partially overlapped components between the seed image and the subsampled image. The halftone-mask image is finally processed by a morphological dilation operation.

The performance of the multiresolution morphology-based text and halftone segmentation method depends upon the residual portions of halftones of an input document image in its corresponding seed image. In a document image, non-text elements (like drawings, maps, graphs, and even halftones) may also be composed of line art. Threshold reduction operations wipe out these types of non-text elements in the corresponding seed image.

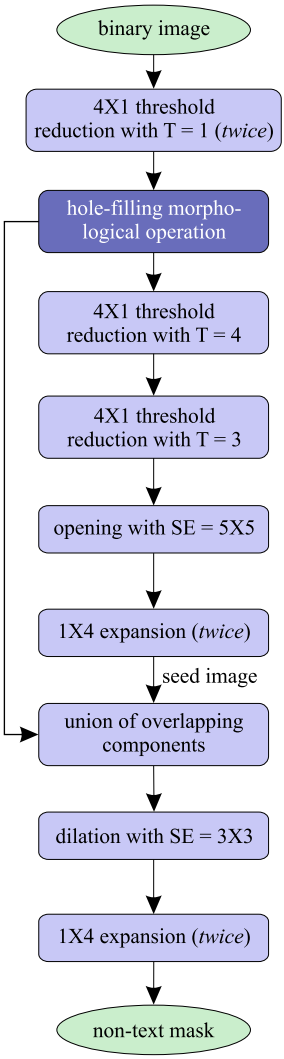
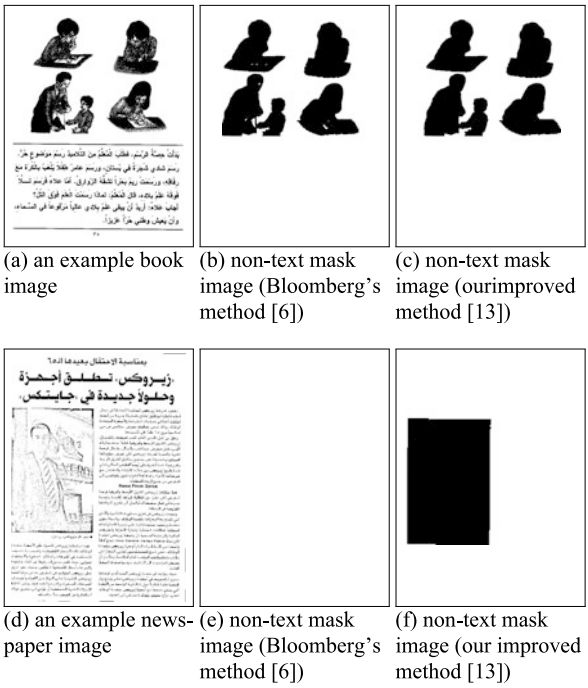


Fig. 2.4 Data flow diagram of improved version of Bloomberg's text and non-text segmentation algorithm [10]. The original Bloomberg's text and non-text algorithm [3] is equivalent to the given data flow diagram without the hole-filling operation. (Note: T: threshold; SE: structuring element)

We introduced an improved version of the multiresolution morphology-based segmentation method that can handle non-text elements like halftones, drawings, and graphics, etc. In the improved version, which is shown in Fig. 2.4, the sub-sampled image is first processed by a hole-filling morphological operation. The hole-filling operation fills drawing type non-text elements, with a better possibility of keeping the residual portions of these non-text elements in the seed image.

Figure 2.5 shows sample Arabic script document images and their text and non-text segmentations for the original version of the multiresolution morphology-based text and non-text segmentation method [3] and its improved version [10]. Our im-



(a) an example book image (b) non-text mask image (Bloomberg's method [6]) (c) non-text mask image (ourimproved method [13])

(d) an example newspaper image (e) non-text mask image (Bloomberg's method [6]) (f) non-text mask image (our improved method [13])

Fig. 2.5 Results of non-text mask using Bloomberg's multiresolution morphology-based text and non-text segmentation method [3] and our improved version [10]. *Top* figure shows a simple case where non-text components are larger than text components and can also be separated by using median size of connected components analysis. *Bottom* figure shows a challenging condition where non-text components are comparable to or even smaller than text components. In contrast to [3], improved multiresolution morphology-based text and non-text segmentation algorithm gives correct result for both simple and challenging conditions

proved version performs well in these examples compared to the original version. In Arabic script document images, like Latin script images, the size of text elements is usually smaller than that of non-text elements, which fits well the assumption of multiresolution morphology-based text and non-text segmentation. Therefore, this approach also works well for Arabic script document images.

2.3 Text Line Detection

Text line detection is an important layout analysis step in document image processing. It is often used before feeding a page to a character recognition engine. The performance of the text line detection operation directly influences the accuracy of the recognition engine.

In the literature, a large number of text line detection approaches are proposed for Arabic document images. Among them, projection profile analysis is a widely used algorithm for detecting text lines in Arabic script document images [22]. It works well for clean document images with large interline spacing, but fails for document images which contain noise, multi-column formats, and small interline spacing. The x–y cut [29] is a state-of-the-art page segmentation approach that is based on project profile analysis. It can handle multi-column documents with small interline spacing. However, it fails for skewed document images and images with large amounts of noise. We presented a ridge-based text line detection approach for

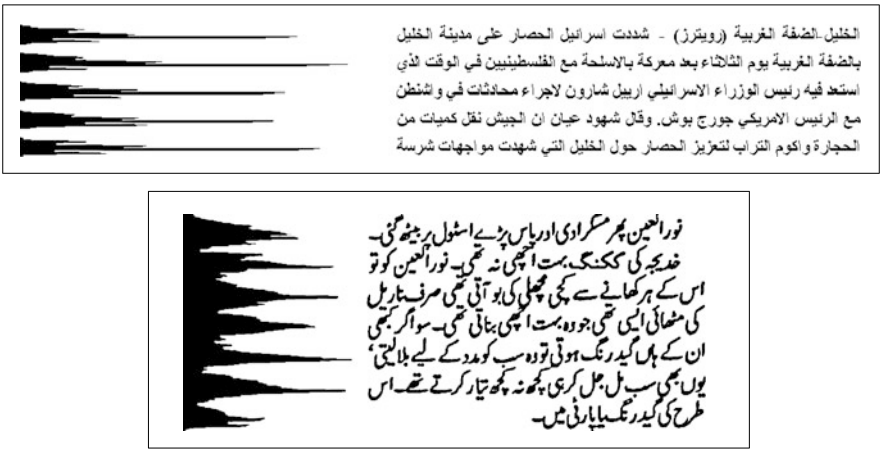


Fig. 2.6 Horizontal projection of Arabic scripts. The *top* figure shows the case of larger, well-defined interline spacing in Naskh script; there are between-line zero valleys in the projection profile. The *bottom* figure shows the case of small interline spacing in Nastaliq script (Urdu); there are no between-line zero valleys in the projection profile

warped camera-captured document images [7, 8]. The ridge-based text line finding method is robust to the presence of noise, skew, and small interline spacing. It can also be used equally for text line detection in different types of Arabic script document images. The $x-y$ cut and ridge-based text line detection methods are described in more detail below.

2.3.1 x–y Cut Text Line Detection Method

The $x-y$ cut page segmentation method [29] is a tree-based algorithm. An input document image is considered as a rectangular block. The $x-y$ cut algorithm recursively cuts a block into smaller blocks, until no block can be cut further. For splitting a block, first its horizontal and vertical projection profiles are computed. The noise removal thresholds t_n^x and t_n^y are then used for computing valleys in the projection profiles. The bins of horizontal and vertical projection profiles are set to zero if they contain values less than linearly scaled threshold t_n^x and t_n^y, respectively, with respect to the width and height of the block. The valleys of the horizontal (v_x) and vertical (v_y) projection profiles are compared with the predefined thresholds t_x and t_y, respectively. The block is split into two blocks at the midpoint of the wider of v_x and v_y, which are larger than t_x and t_y, respectively.

Horizontal projection profiles of sample paragraphs of Naskh and Natsaliq scripts are shown in Fig. 2.6. There are clear zero valleys in the projection profile of the Naskh script corresponding to interline gaps between text lines. In contrast, there is no zero valley in the projection profile of the Nastaliq script. The $x-y$ cut method

(a) Naskh Script (b) X-Y cuts (c) Labeled Text-lines

(d) Nastaliq Script (e) X-Y cuts (f) Labeled Text-lines

Fig. 2.7 The $x-y$ cut algorithm produces correct text line segmentation results for sample Arabic script document images

can be used to segment Nastaliq script documents. In such cases, the noise thresholds are set to a high value for finding the main body of text lines. Afterwards, the remaining portions of text lines are assigned to them through a simple post-processing step. Sample document images of Nastaliq script and their correctly segmented text lines are shown in Fig. 2.7. The following values of thresholds are used for generating these results: $t_n^x = 100$, $t_n^y = 100$, $t_x = 100$, and $t_y = 10$.

The $x-y$ cut algorithm usually fails on documents with a large amount of border noises and reports the whole page as one segment. It also produces wrong text line segmentations for skewed document images. Failed cases of the $x-y$ cut text line segmentation method are shown in Fig. 2.8. The ridge-based text line finding algorithm [7, 8] described next can be used in such cases.

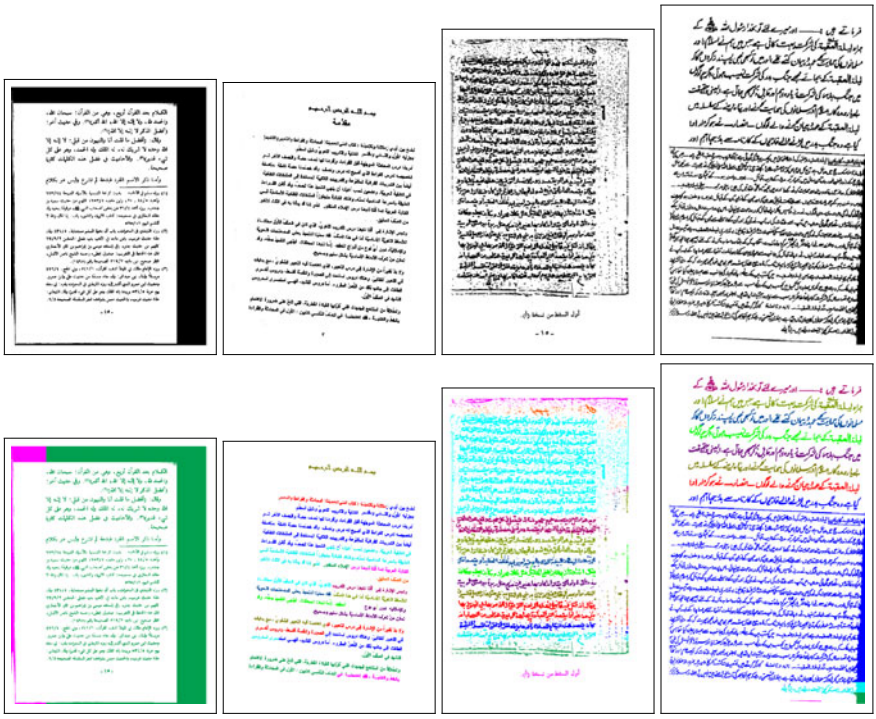


Fig. 2.8 The x–y cut method produces text line segmentation failures for sample document images which contain challenging conditions, like border noise, skew, and a large number of joined characters

2.3.2 Ridge-Based Text Line Detection Method

We introduced a ridge-based text line finding algorithm [7, 8] for warped, Latin script camera-captured document images. The ridge-based text line finding method can be equally applied on different types of document images with respect to digitization methods (scanned or camera-captured), intensity values (binary or grayscale), scripting languages (like Latin, Chinese, Arabic, etc.), and writing styles (typed-text or handwritten). The method consists of two standard image processing techniques: (i) oriented anisotropic Gaussian filter bank smoothing and (ii) ridge detection. A detailed description of the ridge-based text line detection algorithm is presented next.

Step 1: Anisotropic Gaussian Filter Bank Smoothing Here, Gaussian filter bank smoothing is employed for enhancing text line structure in document images, such that it fills intraline gaps and maintains interline spaces. The general formula for an oriented, anisotropic Gaussian filter is shown in Eq. (2.1). It contains three well-defined parameters: σ_x: x-axis standard deviation, σ_y: y-axis standard deviation, and θ: angle of orientation.

$$g(x, y, \sigma_x, \sigma_y, \theta) = \frac{1}{2\pi\sigma_x\sigma_y} \exp\left\{-\frac{1}{2}\left(\frac{(x\cos\theta + y\sin\theta)^2}{{\sigma_x}^2} + \frac{(-x\sin\theta + y\cos\theta)^2}{{\sigma_y}^2}\right)\right\} \tag{2.1}$$

Input: Image I
Output: Smoothed Image I_S
Set $I_S := 0$;
foreach *pixel location* x, y **do**
  for $\sigma_x := w_{\text{start}}$ **to** w_{end} **do**
    for $\sigma_y := h_{\text{start}}$ **to** h_{end} **do**
      for $\theta := \theta_{\text{start}}$ **to** θ_{end} **do**
        $val := g(x, y; \sigma_x, \sigma_y, \theta)$;
        `/* The formula for` $g(x, y, \sigma_x, \sigma_y, \theta)$ `is given in Equation 2.1 */`
        if $val > I_S(x, y)$ **then**
          $I_S(x, y) := val$
        end
      end
    end
  end
end

Algorithm 1: The text line structure enhancement algorithm using multioriented and multiscale anisotropic Gaussian filter bank smoothing

A diverse collection of document images is composed of a wide variety of font sizes, text line orientations, and interline and intraline spaces. Thus, a single isotropic ($\sigma_x = \sigma_y$) or anisotropic ($\sigma_x \neq \sigma_y$) Gaussian filter may either fill interline gaps (for a high value of standard deviation) or leave intraline gaps unfilled (for a small value of standard deviation). In contrast, a set of Gaussian filters, with varying values of standard deviations and orientations, is applied to a document image, and a maximum response is selected for each pixel for the corresponding smoothed image. For generating a filter bank, first the range of values is defined for σ_x, σ_y, and θ. These ranges can either be selected empirically or automatically by analyzing the statistics of connected components in document images. Then, a set of Gaussian filters is generated; each filter is composed of a different combination of values for σ_x, σ_y, and θ. The text line structure enhancement algorithm is shown in Algorithm 1. In order to speed up this algorithm, we used the fast anisotropic Gaussian filter implementation [15, 25]. A sample document image and its corresponding smoothed text line image are shown in Fig. 2.9(a) and 2.9(b), respectively. Now, text lines can be extracted from the smoothed image using the *ridge detection* method, which is described in the second step.

Step 2: Ridge Detection The ridge detection approach is used for representing the shape of objects in digital and speech signal processing. Here, we use the ridge detection approach for finding the main body of text lines in the smoothed document images. Researchers have introduced and analyzed different approaches for

(a) Arabic Nastaliq script (b) Text-line structure enhancement (c) Detected ridges from the smoothed image

(d) Whitespace cuts (e) Detected text-lines regions (f) Color-coded tex-lines labeling

Fig. 2.9 Steps of the ridge-based text line finding algorithm. (**a**) An example image of Arabic Nastaliq script. (**b**) Smoothed text line (text lines enhanced) image, which is generated by using oriented anisotropic Gaussian smoothing filter bank approach. (**c**) Detected ridges from smoothed image using Riley-based ridge detection [33, 34] algorithm. There are over-segmentation errors because of multi-column format. (**d**) Column separators that were detected through whitespace analysis; these separators help in correcting over-segmentation errors. (**e**) Processed ridges using whitespace separators; each ridge covers a complete region of a particular text line. (**f**) *Color-coded labeled text* lines result using detected ridges

ridge detection [13, 14, 26, 34]. Riley [33, 34] introduced the concept of a differential geometry-based ridge detection approach for detecting spectral peeks in speech signals. We use a ridge detection approach that has been derived from Riley's work. An open source version of the ridge detection method is made available as part of the OCRopus OCR system [18]. The Riley-based ridge detection method is described here in detail.

The ridge detection method finds ridge points in an input signal by analyzing its gradient vectors and the greatest downward curvatures. Let us consider a 2D image

I in a Cartesian coordinate system. For a point (x, y), the content of I is represented by $I(x, y)$, and the corresponding gradient vector and the greatest downward curvature are represented by $\nabla I(x, y)$ and $C(x, y)$, respectively. We chose symbol C to represent the greatest downward curvature. The gradient vector $(\nabla I(x, y))$ is defined as:

$$\nabla I(x, y) = \left(\frac{\partial I(x, y)}{\partial x}, \frac{\partial I(x, y)}{\partial y} \right) \tag{2.2}$$

and the greatest downward curvature $(C(x, y))$ is defined as:

$$C(x, y) = \frac{\mathbf{e}_s(x, y)}{|\mathbf{e}_s(x, y)|}, \tag{2.3}$$

where $\mathbf{e}_s(x, y)$ is the eigenvector of the small eigenvalue of the Hessian matrix at point (x, y); $\lambda_s(x, y)$ represents the small eigenvalue and $\lambda_l(x, y)$ represents the large eigenvalue. Each point in image I is analyzed by Eq. (2.4) for checking whether it is a ridge point or not.

$$R(x, y) = \begin{cases} 1 & \text{if } \begin{cases} 1.\ \lambda_s(x, y) < 0 \text{ and} \\ 2.\ (\nabla I(x, y) \cdot C(x, y)) = 0 \end{cases} \\ 0 & \text{else} \end{cases} \tag{2.4}$$

The condition in Eq. (2.4) is sufficient for finding ridges in continuous signals. In order to make it applicable for discrete signals, more tests are needed for analyzing whether or not a point is a ridge's point. The ridge detection method that we have used here is derived from Riley's work of ridge detection for discrete signals. In this case, each point is analyzed for a ridge's point with each of its neighboring points based on the rules mentioned in Eq. (2.5), and is considered a ridge point if the output of Eq. (2.6) is 1. A complete description of the ridge detection method is shown in Algorithm 2.

$$R(x, y, dx, dy) = \begin{cases} 1 & \text{if } \begin{cases} 1.\ \lambda_s(x, y) < 0 \text{ and } |\lambda_s(x, y)| > |\lambda_l(x, y)| \text{ and} \\ 2.\ \lambda_s(x + dx, y + dy) < 0 \text{ and} \\ \quad |\lambda_s(x + dx, y + dy)| > |\lambda_l(x + dx, y + dy)| \text{ and} \\ 3.\ \nabla I(x, y) \cdot \nabla I(x + dx, y + dy) < C(x, y) \\ \quad \cdot C(x + dx, y + dy) \text{ and} \\ 4.\ (\nabla I(x, y) \cdot C(x, y))(\nabla I(x + dx, y + dy) \\ \quad \cdot C(x + dx, y + dy)) \\ \quad (C(x, y) \cdot C(x + dx, y + dy)) < 0 \end{cases} \\ 0 & \text{else} \end{cases} \tag{2.5}$$

$$R(x, y) = \max_{(dx,dy) \in (0,1),(1,0),(0,-1),(-1,0)} R(x, y, dx, dy) \tag{2.6}$$

The ridge detection output for the smoothed image of Fig. 2.9(b) is shown in Fig. 2.9(c). A ridge covers a complete region of a particular text line for single-column document images. For multi-column documents, the filter bank may fill small gaps between text lines between different columns. Therefore, a single ridge may cover either a single text line or multiple text lines at the same height (Fig. 2.9(c)). This situation causes over-segmentation errors. This type of over-segmentation error can be corrected by a *whitespace analysis*, as described below.

Input: The Smoothed Image I
Output: Detected Ridges Image I_R
Calculate gradient vectors image ∇I;
Calculate greatest downward curvature image $C : \mathbf{e}_s, \lambda_s, \mathbf{e}_l, \lambda_l$;
Set $I_R := 0$;
foreach *pixel location* x, y **do**
  if $R(x, y, 0, 1)$ *or* $R(x, y, 1, 0)$ *or* $R(x, y, 0, -1)$ *or* $R(x, y, -1, 0)$
  then `/* The formula for` $R(x, y, dx, dy)$ `is given in Eq. (2.5) */`
    $I_R(x, y) = 1$
  else
    $I_R(x, y) = 0$
  end
end

Algorithm 2: The Riley [33, 34]-based ridge detection algorithm

Step 3: Whitespace Analysis Whitespace analysis aims to find a set of maximal white rectangles in a document image such that the union of these rectangles completely covers the document's background. It is usually used for page segmentation [2] or multi-column separation [5]. An algorithm for finding maximal whitespace rectangles is presented in [5]. The main idea behind that algorithm is similar to the quick-sort or branch-and-bound methods. The whitespace rectangles are evaluated as candidates for column separators or gutters based on their statistics, like aspect ratio, width, etc. Whitespace cuts that correspond to column separators are shown in Fig. 2.9(d). Now, over-segmentation errors (as shown in Fig. 2.9(c)) can be corrected by cutting detected ridges at those points which lie over whitespace rectangles. The output ridges are shown in Fig. 2.9(e), where a single ridge covers a single text line. These ridges are considered as detected text lines.

Step 4: Text Line Labeling A text line labeling method assigns a unique label to all of the connected components of a text line. As shown in Fig. 2.9(e), each detected ridge represents the main region of a particular text line. Let us consider a simple case where each connected component in a document image overlaps with a single ridge. First, each ridge is assigned a unique label. Then, each connected component is assigned the label of its corresponding ridge. Each unlabeled connected component in the proximity of text lines is assigned the label of its nearest text line. It is also possible that a connected component overlaps with more than one ridge, which usually happens in the case of inter-line touching or overlapping. In such a case, a connected component is cut in the center of each pair of consecutive ridges, and then each portion is assigned the label of the particular ridge. The result of text line labeling in color-coded form is shown in Fig. 2.9(f).

For some of the challenging problems like document skew and noise, the text line detection results of the ridge-based text line extraction method are shown in Fig. 2.10. As can be seen in the figure, the ridge-based text line detection method is robust to document skew, small interline gaps, border noise, and interline touching and/or overlapping as compared to the x–y cut method (whose results are shown in Fig. 2.8).

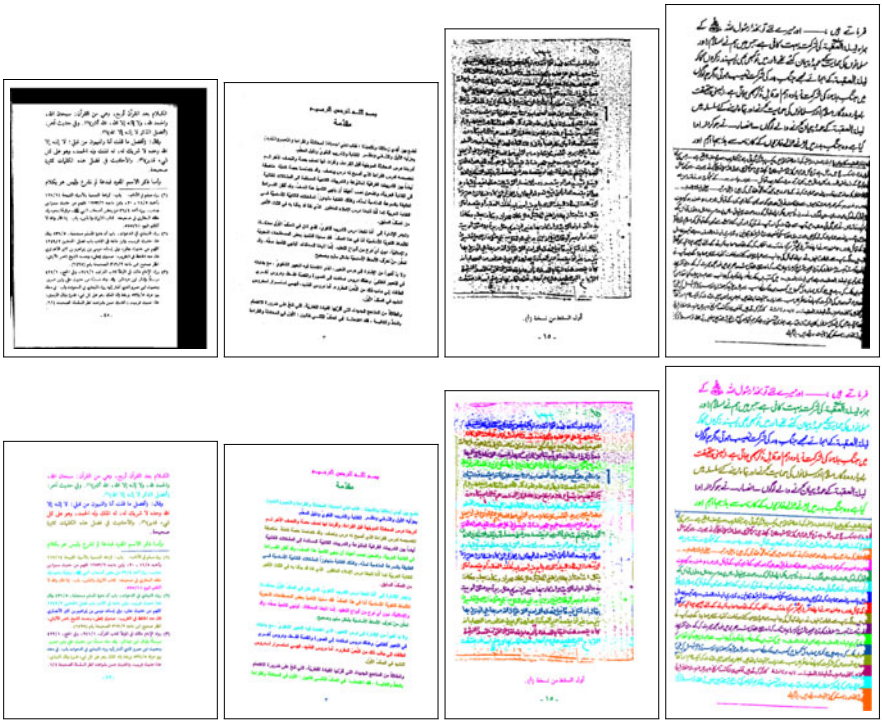


Fig. 2.10 The ridge-based text line finding method produces correct text line segmentation results for sample document images with border noise, skew, and a large number of joined characters

2.4 Text Line Reading Order Determination

A reading order determination method tries to find the order of text lines with respect to their corresponding reading flow. The reading order can be determined by applying some ordering criteria over the positioning of the text lines. In contrast to Latin script, the reading order of Arabic script is from right to left. A reading order determination method is presented in [6] for Latin script, which is modified for Nastaliq Arabic script in [36]. The ordering criteria that were presented in [36] are stated as follows:

- "Text-Line 'a' comes before text-line 'b' if their ranges of x-coordinates overlap and if text-line 'a' is above text-line 'b' on the page".
- "Text-Line 'a' comes before text-line 'b' if a is entirely to the right of 'b' and if there does not exist a text-line 'c' whose y-coordinates are between 'a' and 'b' and whose range of x-coordinates overlaps both 'a' and 'b' ".

The reading order determination method [6, 36] finds the partial ordering of text lines through the above-defined ordering criteria, and then finds a complete order using a topological sorting algorithm [12].

Fig. 2.11 Example images illustrating results of reading order for a newspaper and a book page. *Thin horizontal lines* with different colors indicate detected text line segments, and the *magenta lines* running down and diagonally across the image indicate reading order

Examples of reading order determination on sample document images are shown in Fig. 2.11. The performance of the reading order determination method decreases with a decrease in text line detection accuracy and/or the presence of noise in document images.

2.5 Discussion

In this chapter, we have presented a generic layout analysis system which can be used for a variety of typed-text (like Naskh and Nastaliq), handwritten, and ancient Arabic script document images. Our layout analysis system first performs text and non-text segmentation, then text line detection, and finally reading order determination. The output of our layout analysis system in conjunction with an efficient OCR engine can be used for the digitization of a wide variety of typed-text Arabic script document images.

We have demonstrated that the improved version of the multiresolution morphology-based text and non-text segmentation approach is suitable for Arabic script

document images. It gives correct text and non-text segmentation results, unless a document image contains very large text element(s) and/or very small non-text element(s).

The projection profile-based $x-y$ cut method gives correct text line segmentation results for clean document images. It produces bad segmentation results for document images containing large amounts of noise and/or document skew. We have proposed a robust text line detection approach using standard image processing methods (filter bank Gaussian smoothing and ridge detection) for Arabic script document images. Our ridge-based text line finding approach is robust to large amounts of border noises, handwritten marks, and document skew and curl and is equally applicable on both binary and grayscale document images. It has well-understood free parameters, such as ranges of orientation angles, and x- and y-axis standard deviations for oriented anisotropic Gaussian filter bank smoothing. These parameter values can be easily tuned for a variety of document images with respect to the size statistics of connected components.

The performance of the reading order determination algorithm heavily depends on the text line detection accuracy. Manual inspection of the results showed the following types of errors: (1) if two text lines from different text columns are merged, they are interpreted as a separator, and hence the algorithm gives a wrong reading order, and (2) in some cases, the separation between different sections of a multi-column document is not represented by a text line spanning the columns, but instead a ruling (thick horizontal black line) is used. In that case the algorithm fails to detect the start of a new section.

In general, our generic layout analysis system is quite robust to different types of challenging problems in complex document image layouts, like books, newspapers, and ancient Arabic script document images. To the best of our knowledge, there is no public dataset available for the evaluation of different layout analysis systems for Arabic document images. As a future goal, there is a need for a public dataset with a large variety of Arabic script documents with text line level ground truth for benchmarking existing layout analysis systems for Arabic scripts.

References

1. Abed, H.E., Märgner, V.: ICDAR 2009—Arabic handwriting recognition competition. Int. J. Doc. Anal. Recognit. **14**, 3–13 (2011)
2. Baird, H.S.: Background structure in document images. In: Bunke, H., Wang, P., Baird, H.S. (eds.) Document Image Analysis, pp. 17–34. World Scientific, Singapore (1994)
3. Bloomberg, D.S.: Multiresolution morphological approach to document image analysis. In: Proceedings 1st International Conference on Document Analysis and Recognition, St. Malo, France, pp. 963–971 (1991)
4. Boussellaa, W., Zahour, A., Abed, H.E., Benabdelhafid, A., Alimi, A.M.: Unsupervised block covering analysis for text-line segmentation of Arabic ancient handwritten document images. In: Proceedings 20th International Conference on Pattern Recognition, Istanbul, Turkey, pp. 1929–1932 (2010)
5. Breuel, T.M.: Two geometric algorithms for layout analysis. In: Proceedings Document Analysis Systems, Princeton, NY, USA, Aug. 2002, pp. 188–199 (2002)

6. Breuel, T.M.: High performance document layout analysis. In: Symposium on Document Image Understanding Technology, Greenbelt, MD, USA, April 2003
7. Bukhari, S.S., Shafait, F., Breuel, T.M.: Ridges based curled textline region detection from grayscale camera-captured document images. In: Computer Analysis of Images and Patterns. Lecture Notes in Computer Science, vol. 5702, pp. 173–180 (2009)
8. Bukhari, S.S., Shafait, F., Breuel, T.M.: Script-independent handwritten textlines segmentation using active contours. In: Proceedings 10th International Conference on Document Analysis and Recognition, Barcelona, Spain, pp. 446–450 (2009)
9. Bukhari, S.S., Shafait, F., Breuel, T.M.: Document image segmentation using discriminative learning over connected components. In: Proceedings 9th IAPR Workshop on Document Analysis Systems, Boston, Massachusetts, USA, pp. 183–190 (2010)
10. Bukhari, S.S., Shafait, F., Breuel, T.M.: Improved document image segmentation algorithm using multiresolution morphology. In: Proceedings SPIE Document Recognition and Retrieval XVIII, San Jose, CA, USA, Jan. 2011
11. Cattoni, R., Coianiz, T., Messelodi, S., Modena, C.M.: Geometric layout analysis techniques for document image understanding: a review. Technical Report 9703-09, IRST, Trento, Italy (1998)
12. Cormen, T.H., Leiserson, C.E., Rivest, R.L.: Introduction to Algorithms, pp. 485–488. MIT Press, Cambridge (1990). Chapter 23
13. Damon, J.: Properties of ridges and cores for two-dimensional images. J. Math. Imaging Vis. **10**, 163–174 (1999)
14. Eberly, D., Gardner, R., Morse, B., Pizer, S., Scharlach, C.: Ridges for image analysis. J. Math. Imaging Vis. **4**, 353–373 (1994)
15. Geusebroek, J.M., Smeulders, A.W.M., Weijer, J.V.D.: Fast anisotropic Gauss filtering. IEEE Trans. Image Process. **12**, 2003 (2003)
16. http://en.wikipedia.org/wiki/Nastaliq_script
17. http://www.ijma3.org/Templates/InsideTemplate.aspx?PostingId=427
18. http://code.google.com/p/ocropus/
19. Husain, S.A., Amin, S.H.: A multi-tier holistic approach for Urdu Nastaliq recognition. In: IEEE International Multi-topic Conference, Karachi, Pakistan, Dec. 2002
20. Jelodar, M.S., Fadaeieslam, M.J., Mozayani, N., Fazeli, M.: A Persian OCR system using morphological operators. World Acad. Sci., Eng. Technol. **4**, 141 (2007)
21. Keysers, D., Shafait, F., Breuel, T.M.: Document image zone classification—a simple high-performance approach. In: 2nd International Conference on Computer Vision Theory and Applications, Barcelona, Spain, Mar. 2007, pp. 44–51 (2007)
22. Khorsheed, M.S.: Off-line Arabic character recognition—a review. Pattern Anal. Appl. **5**(1), 31–45 (2002)
23. Kise, K., Sato, A., Iwata, M.: Segmentation of page images using the area Voronoi diagram. Comput. Vis. Image Underst. **70**(3), 370–382 (1998)
24. Kumar, K.S., Kumar, S., Jawahar, C.: On segmentation of documents in complex scripts. In: 9th International Conference on Document Analysis and Recognition, Curitiba, Brazil, Sep. 2007, pp. 1243–1247 (2007)
25. Lampert, C.H., Wirjadi, O.: An optimal nonorthogonal separation of the anisotropic Gaussian convolution filter. IEEE Trans. Image Process. **15**(11), 3501–3513 (2006)
26. Lindeberg, T.: Edge detection and ridge detection with automatic scale selection. Int. J. Comput. Vis. **30**, 465–470 (1996)
27. Moll, M.A., Baird, H.S., An, C.: Truthing for pixel-accurate segmentation. In: Proceedings 8th IAPR International Workshop on Document Analysis Systems, Sep. 2008, pp. 379–385 (2008)
28. Nagy, G.: Twenty years of document image analysis in PAMI. IEEE Trans. Pattern Anal. Mach. Intell. **22**(1), 38–62 (2000)
29. Nagy, G., Seth, S., Viswanathan, M.: A prototype document image analysis system for technical journals. Computer **7**(25), 10–22 (1992)

30. O'Gorman, L.: The document spectrum for page layout analysis. IEEE Trans. Pattern Anal. Mach. Intell. **15**(11), 1162–1173 (1993)
31. Otsu, N.: A threshold selection method from gray-level histograms. IEEE Trans. Syst. Man Cybern. **9**(1), 62–66 (1979)
32. Pal, U., Sarkar, A.: Recognition of printed Urdu script. In: 7th International Conference on Document Analysis and Recognition, Edinburgh, UK, Aug. 2003, pp. 1183–1187 (2003)
33. Riley, M.D.: Beyond quasi-stationarity: Designing time-frequency representations for speech signals. In: Proceedings IEEE International Conference on Acoustics, Speech, and Signal Processing, Apr. 1987, vol. 12, pp. 657–660 (1987)
34. Riley, M.D.: Time-frequency representation for speech signals. Ph.D. Thesis, MIT (1987)
35. Sauvola, J., Pietikainen, M.: Adaptive document image binarization. Pattern Recognit. **33**(2), 225–236 (2000)
36. Shafait, F., Hasan, A., Keysers, D., Breuel, T.M.: Layout analysis of Urdu document images. In: 10th IEEE International Multi-topic Conference, INMIC'06. Islamabad, Pakistan, Dec. 2006
37. Shafait, F., Keysers, D., Breuel, T.M.: Performance evaluation and benchmarking of six page segmentation algorithms. IEEE Trans. Pattern Anal. Mach. Intell. **30**(6) (2008)
38. Shirali-Shahreza, S., Manzuri-Shalmani, M.T., Shirali-Shahreza, M.H.: Page segmentation of Persian/Arabic printed text using ink spread effect. In: SICE-ICASE International Joint Conference, Busan, Korea, Oct. 2006, pp. 259–262 (2006)
39. Wang, Y., Haralick, R., Phillips, I.: Improvement of zone content classification by using background analysis. In: Proceedings Document Analysis Systems, Rio de Janeiro, Brazil, Dec. 2000
40. Wong, K.Y., Casey, R.G., Wahl, F.M.: Document analysis system. IBM J. Res. Dev. **26**(6), 647–656 (1982)
41. Zahour, A., Taconet, B., Mercy, P., Ramdane, S.: Arabic hand-written text-line extraction. In: Proceedings 6th International Conference on Document Analysis and Recognition, Seattle, WA, USA, Sep. 2001, pp. 281–285 (2001)

http://www.springer.com/978-1-4471-4071-9

Guide to OCR for Arabic Scripts
Märgner, V.; El Abed, H. (Eds.)
2012, XX, 592 p., Hardcover
ISBN: 978-1-4471-4071-9