Pattern Recognition 44 (2011) 917–928

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

A new scheme for unconstrained handwritten text-line segmentation

Alireza Alaei [a,*], Umapada Pal [b], P. Nagabhushan [a]

[a] Department of Studies in Computer Science, University of Mysore, Mysore 570 006, India
[b] Computer Vision and Pattern Recognition Unit, Indian Statistical Institute, Kolkata 700108, India

ARTICLE INFO

Article history:
Received 12 December 2009
Received in revised form
15 October 2010
Accepted 24 October 2010

Keywords:
Handwritten document processing
Unconstrained handwritten line segmentation
Piece-wise painting
Mathematical morphology
Thinning

ABSTRACT

Variations in inter-line gaps and skewed or curled text-lines are some of the challenging issues in segmentation of handwritten text-lines. Moreover, overlapping and touching text-lines that frequently appear in unconstrained handwritten text documents significantly increase segmentation complexities. In this paper, we propose a novel approach for unconstrained handwritten text-line segmentation. A new painting technique is employed to smear the foreground portion of the document image. The painting technique enhances the separability between the foreground and background portions enabling easy detection of text-lines. A dilation operation is employed on the foreground portion of the painted image to obtain a single component for each text-line. Thinning of the background portion of the dilated image and subsequently some trimming operations are performed to obtain a number of separating lines, called candidate line separators. By using the starting and ending points of the candidate line separators and analyzing the distances among them, related candidate line separators are connected to obtain segmented text-lines. Furthermore, the problems of overlapping and touching components are addressed using some novel techniques. We tested the proposed scheme on text-pages of English, French, German, Greek, Persian, Oriya and Bangla and remarkable results were obtained.

1. Introduction

Text-line segmentation is an important step towards the automatic recognition of off-line handwritten text documents. Variations in inter-line distance, presence of inconsistent baseline skew, and touching and overlapping text-lines make this task more crucial and complex. Correctness/incorrectness of text-line segmentation directly affects the accuracy of word/character segmentation, which consequently changes the accuracy of word/character recognition [10]. Several techniques for text-line segmentation are reported in the literature [1–18]. These techniques may be categorized into four groups [10,13] as follows: (i) projection profile based techniques, (ii) Hough transform based techniques, (iii) smearing techniques and (iv) methods based on thinning operations.

As a conventional technique for text-line segmentation, global horizontal projection analysis of black pixels has been utilized in [1,2]. However, this technique cannot be used directly on unconstrained handwritten text documents due to text-line skew variability, inconsistent inter-line distances and overlapping and touching components of two consecutive text-lines.

Partial or piece-wise horizontal projection analysis of black pixels as a modification of the global projection technique is employed by many researchers [3–7] to segment handwritten text-pages of different languages. In the piece-wise horizontal projection technique, an image is decomposed into vertical stripes and the horizontal projection of each stripe is computed. The positions of potential piece-wise separating lines are obtained either by applying heuristics on the partial projections or by formulating an HMM model based on the statistics of candidate text and gap areas [7]. The potential separating lines are then connected to segment text-lines. The overlapping and touching cases are then treated either by applying a crossing technique, a contour tracing algorithm [3–5] or by exploiting bivariate Gaussian densities to represent the text-lines.

These piece-wise projection based methods [3–7] have a few shortcomings: (a) they generate too many potential separating lines, (b) the parameter of stripe width is predefined, (c) text-lines should not have significant skew as mentioned in [8] and (d) if there is no potential piece-wise line in the first and last stripes, drawing a complete separating line will become impossible in the algorithms presented in [3–5]. These shortcomings are observed based on an experiment that we have conducted with a number of text-pages. Some authors [4,5] also used skew information for text-line separation. In an unconstrained handwritten text-page, it is very difficult to detect the orientation of each line on the basis of the skew calculated for the entire page. Therefore, these methods may not work properly.

The concept of the Hough transform is employed in the field of document analysis for many purposes such as skew detection, line detection, slant detection and text-line segmentation [13]. In [11–13],

* Corresponding author.
E-mail addresses: alireza20alaei@yahoo.com (A. Alaei), umapada@isical.ac.in (U. Pal), pnagabhushan@hotmail.com (P. Nagabhushan).

0031-3203/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.patcog.2010.10.014

the Hough transform is employed for text-line segmentation in different scripts. In [12,13], a block-based Hough transform is presented which is a modification of the conventional Hough transform methodology. The algorithm includes partitioning of the connected component domain into three spatial sub-domains and applying a block-based Hough transform to detect the potential text lines.

Under the smearing methods, the fuzzy Run Length Smoothing Algorithm (RLSA) [14] and improved directional run-length analysis [15] are employed. In [14], the fuzzy RLSA measure is calculated for every pixel in the input image. A new gray-scale image is created based on the RLSA measure and then the image is binarized. The lines of the text are extracted from the binarized image. In [15], by applying the RLSA measure, words are smeared and then the foreground portion of the modified image is eroded. The erosion is also performed on background portions to obtain boundary information of the text-lines. Text-lines are segmented by using positional information obtained from the foreground and the boundary information.

A thinning operation has also been used by other researchers for text-line segmentation of Japanese and Indian text documents [15,18]. In [18], a thinning algorithm followed by post-processing operations are employed for the entire background region of an input text image to detect the separating borderlines.

Recently, some techniques have used level set, active contour and a variational Bayes method for text-line segmentation [8,16,17]. In [8], density estimation and the level set method were utilized for text-line segmentation. A probability map is estimated from an input document image, where each element represents the probability of the original pixel belonging to a text line. The level set method is utilized to determine the boundary evolution of neighboring text lines. In [16], at first, a matched filter bank approach is used for smoothing the input text image. The central line of text-line components is then computed using ridges over the smoothed image. Finally, the active contours (snakes) over the ridges are adapted to obtain the text-line segmentation result. In [17], the variational Bayes method was employed for text-line segmentation of Chinese handwritten texts. A document image was considered as a Gaussian mixture model, with each component corresponding to a text line. To estimate the density parameters and determine the number of text lines, the variational Bayes framework was exploited.

Documents written in Greek, Persian and Arabic contain abundant dots, strokes and diacritics. In such documents, a connected component can be a whole word, a sub-word, a character, a dot, a component of connected dots, or a stroke. The presence of dots, strokes and diacritics makes the segmentation problem more challenging and some of the techniques (such as the Hough transform-based approach [11], improved directional run-length smearing based techniques [15] and the active contour analysis based method [16]) proposed in the literature may fail for these types of documents. The reasons are that: (i) the Hough transform-based approach [11] considers as equally important a whole word and a small dot/stroke in the Hough domain, (ii) the run-length based smearing technique [15] may connect dots with some strokes and such connected components may be segmented as a separate text-line and (iii) use of a filter bank in [16] for smoothing may remove dot(s) and strokes from the documents. Dots and strokes play an important role in Greek, Persian and Arabic documents and deletion of such dots and strokes from documents sometimes generates improper segmentation of text-lines. Our proposed approach uses a novel piece-wise painting concept where such dots and strokes are painted and they will be attached to the main part of the text-line for proper segmentation.

In the proposed algorithm, initially the document image is vertically split. Determination of the appropriate width for stripping is done using the novel painting technique and based on gaps between text-lines in an input image. Employing the painting and smoothing operations the foreground information is then smeared in a stripe-wise fashion instead of smearing the whole image at a time. As a result in a text-line, each piece of smeared foreground may be connected to one or two adjacent pieces of smeared foreground. This connectivity preserves the skew of each text-line and the problem of skew variations in text-lines cannot affect our method. In the next step, a dilation operation is applied on the smoothed image to get a connected component for each text-line. To get the appropriate separating line, a thinning algorithm is applied only on the background portions of the dilated image. Since a thinning algorithm is used on the background portions of the dilated image, we work on a small set of candidate line separators instead of dealing with too many piece-wise lines. By trimming the candidate line separators and based on some heuristic rules, text-line segmentation is carried out. Finally, the touching and overlapping situations are addressed.

The organization of the rest of the research paper is as follows: Section 2 contains the description of the proposed text-line segmentation methodology. Experimental results and performance analysis are discussed in Section 3. In the last section, conclusions and future work are put forward.

2. Proposed text-line segmentation approach

The block diagram of the proposed text-line segmentation algorithm is shown in Fig. 1, which depicts different stages clearly. The stages are: (a) Piece-wise Painting Algorithm (PPA), (b) dilation operation, (c) complementing the dilated image followed by a thinning operation, (d) trimming the extracted lines, (e) constructing separating lines and (f) resolving the problems of overlapping and touching components. Each stage is described in detail in the following sub-sections.

2.1. Piece-wise Painting Algorithm (PPA)

The Piece-wise Painting Algorithm (PPA) starts by dividing the image into a number of vertical segments, called stripes. In our work, stripes are considered from left to right. In the research work presented in [4], the width of the stripe (SW) is considered as the average size of words in Arabic (5 characters). In [6], the width of a stripe is considered as 5% of the text-image width. However, with a clear observation it was noticed that the width of the stripe should actually be dependent on the gaps (white space) between the text-lines and it should be dynamically calculated for each text-page. In other words, if the text-lines are closely written then the width of the stripe should be very small and vice versa. Initially, the input text-image is decomposed into vertical stripes based on the average width of the connected components (see Fig. 2). In a noisy dataset or in the presence of many broken characters in text-pages of a dataset, the use of a median filter is applicable to reduce the affect of very small components in order to get reliable results. Since the image is divided from the left to right direction, the width of the last stripe may become smaller than the other stripes. From an experimental study, we noted that the direction of traversing an input text image (from left to right or from right to left) will not affect the performance of the proposed algorithm.

Subsequent to the division of the input image into stripes, the gray value of each pixel in each row of a stripe is modified by changing it with the average gray value of all pixels present in that row of the stripe. The average gray value (GLM) in each row-stripe is computed using the following formula:

$$GLM_{ik} = \sum_{j} I_{ij}/SW_k \quad (1)$$

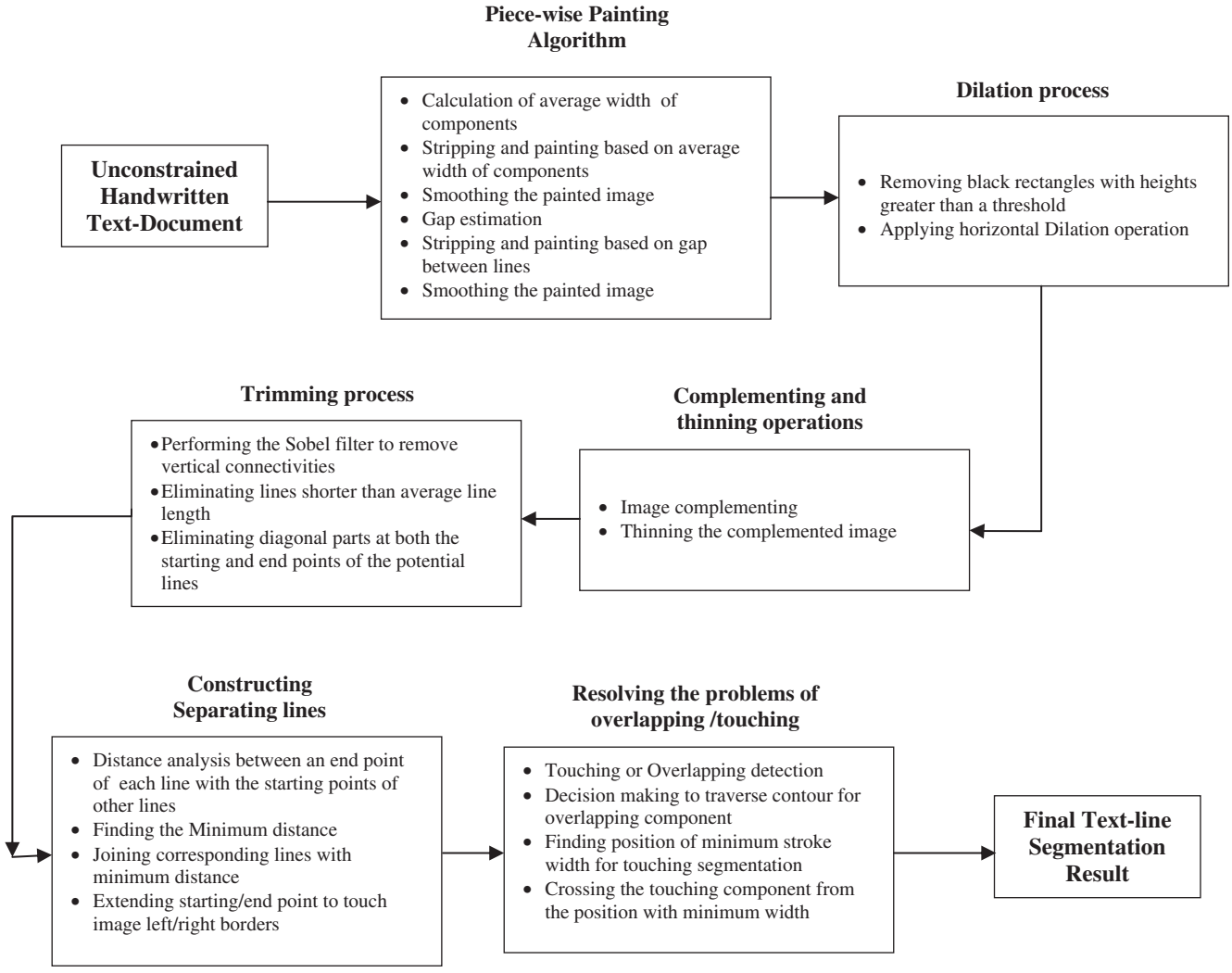

Fig. 1. Block diagram of the proposed method.

where $k=1$ to no. of stripe, $j=(k-1)\times SW_k+1$ to $k\times SW_k$, $i=$row number.

The GLM_{ik} is the average gray value of all the pixels placed in the ith row and kth stripe, I_{ij} is the gray value in the ith row and jth column of the input image I and SW_k is the width of kth stripe. This is the newly introduced concept of painting a stripe and the entire text image is painted accordingly. It may be noted that if the input text-image is a binary one then it is converted into a gray-scale image employing 2×2 mean filtering 5 times. To choose mask size and number of iteration, we analyzed the results obtained from different masks and iterations. Based on this analysis, we noted that 2×2 mask generated better results with lesser time when filtering was applied 5 times. In the case of a gray-scale image with a complex non-uniform background, a sophisticated binarization technique such as [22] is applicable to obtain a better segmentation result.

The resultant gray-scale image is converted to binary by applying the Otsu's method [23] on each stripe separately (see Fig. 3). The black and white rectangles represent the foreground (text regions) and background, respectively. This image is smoothed stripe-wise by a set of smoothing operations to fill white space between two consecutive black regions by black (connecting two consecutive black regions) and to remove very thin black areas (obtained from dots, strokes, descenders and ascenders of characters) by converting them to white. This is done to avoid having redundant segmentation lines that may produce improper text-line segmentation. Therefore, heights of all the white rectangles in each stripe are computed and the statistical mode of the heights in each stripe is calculated. The white space (white rectangle) between two consecutive black regions is filled by black pixels based on the statistical mode MW_k using the following criterion:

$$\text{if } HW_{ik} < MW_k \text{ then } VW_{ik} = 0 \tag{2}$$

where $k=1$ to no. of stripes, $i=1$ to no. of white areas in kth stripe. HW_{ik} denotes the height of the ith white region in the kth stripe, MW_k denotes the statistical mode of heights of white areas in the kth stripe and $VW_{ik}\in\{0,1\}$ is the value of ith white area in the kth stripe (0 for black and 1 for white).

The black thin rectangles and dangling black rectangles (black rectangles that do not touch any other black rectangles in the next/previous stripe) are converted into white based on the following criterion:

$$\begin{cases} \text{if } HB_{ik} < T & \text{then } VB_{ik} = 1 \\ \text{if}(HB_{ik} < 3\times T \ \&\ HB_{ik} \text{ is dangling}) & \text{then } VB_{ik} = 1 \end{cases} \tag{3}$$

where HB_{ik} denotes the height of the ith black region in the kth stripe, T is a threshold obtained from the heights of black areas, and $VB_{ik}\in\{0,1\}$ is the value of the ith black area in the kth stripe

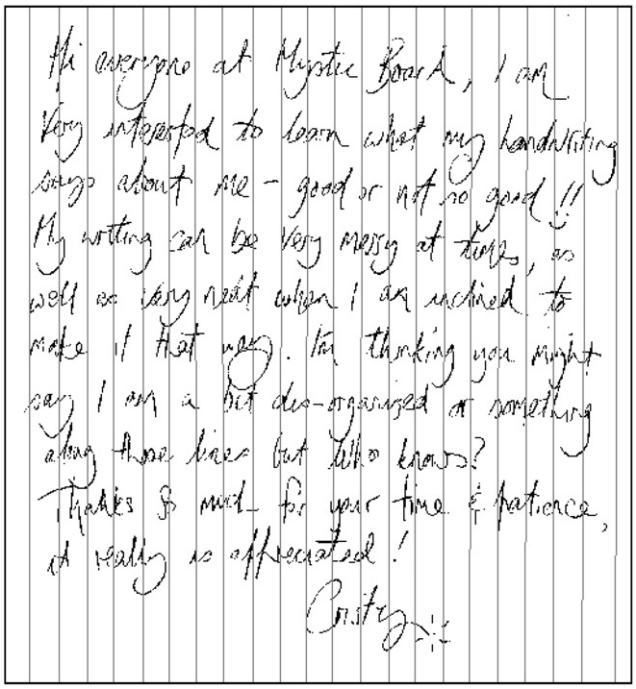

Fig. 2. Decomposition of an input image into stripes based on average width of components.

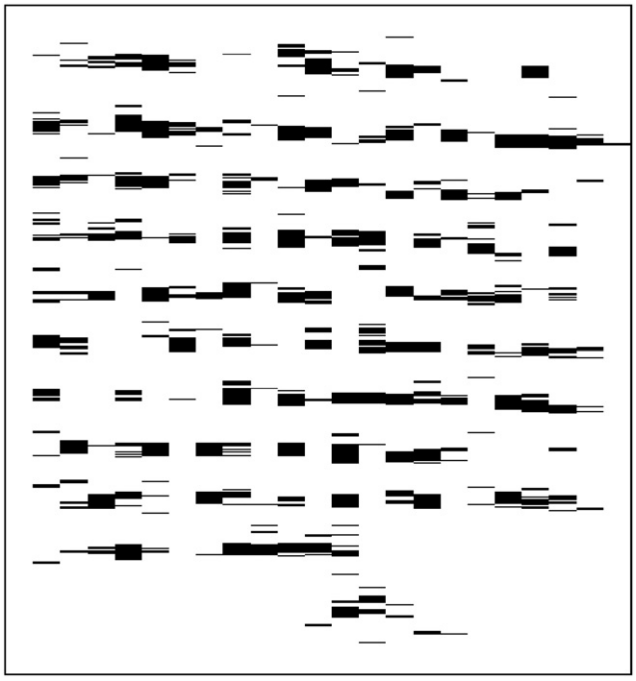

Fig. 3. Text-page after applying the Painting Algorithm on Fig. 2.

(0 for black and 1 for white). The threshold T is dynamically computed with respect to each input image. To do so, stripe-wise heights of all the black rectangles (smeared foreground information as shown in Fig. 3) obtained from the Painting Algorithm were sorted in ascending order. Then, we divided the sorted list into 2, 4, 6, 8, 10 and 16 parts and defined T as the mean value of the first part of each of these divisions. The performance values for each value of T were 93.21%, 94.58%, 96.08%, 96.57%, 95.498% and 95.498%, respectively. Therefore, we conclude experimentally that 8 is the most efficient value. For the experiment, we considered 100 images from three different handwritten databases (first 35 images of Ref. [9], first 35 images of Ref. [13] and first 30 images of our dataset). The resultant smoothed image obtained from Fig. 3 is shown in Fig. 4.

In the next step, the statistical mode of the heights of all white rectangles of the smoothed image is computed. Based on this value, the input image is divided into stripes again (Fig. 5). The proposed painting technique and the smoothing algorithm are applied once more. The results are presented in Figs. 6 and 7, respectively.

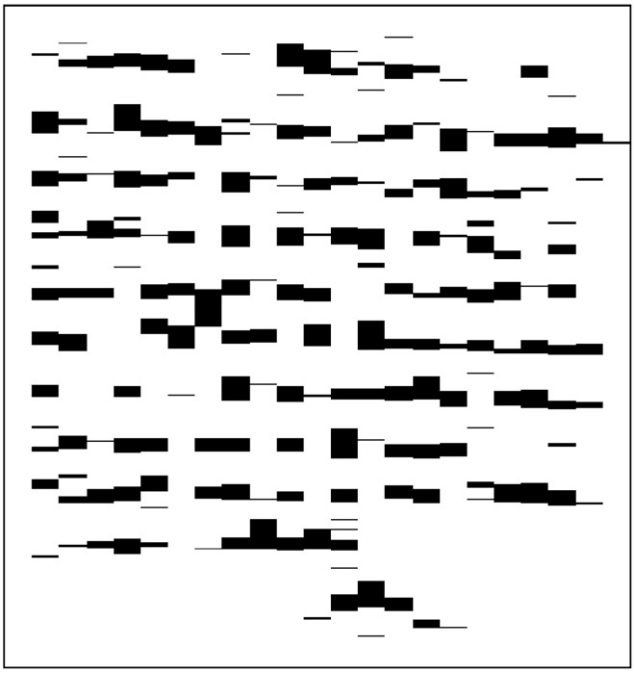

Fig. 4. Results obtained after filling the white space between two black rectangles and removing black rectangles of small heights from Fig. 3.

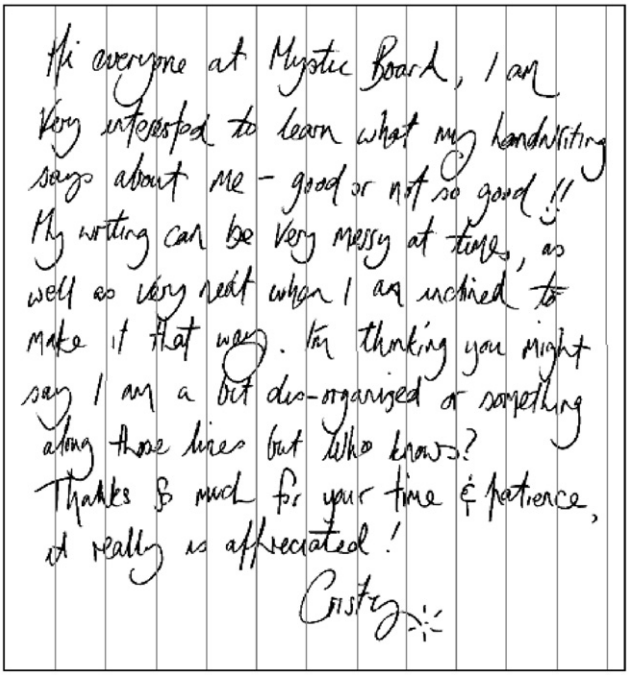

Fig. 5. Original text-page after decomposing it into new stripes.

Fig. 6. Results after applying the Painting Algorithm (gray image) on Fig. 5.

Fig. 7. Text-page after filling the white space between two black rectangles and removing black rectangles of small heights (the filled in portions are shown in gray).

2.2. Dilation operation

There may be disconnectivities in some lines of the smoothed image. Also, some rectangular black areas may appear that connect two consecutive text-lines or have longer heights with respect to the heights of the other rectangular black areas. These areas indicate overlapping/touching portions. Generally, when two text-lines touch/overlap, two or more consecutive rectangular black areas obtained from those two text-lines are joined and make a longer black area. The height of this rectangular black area will be greater or equal to twice the size of the average height of the black areas. To get an idea about the heights of different rectangular black areas, a bar chart is displayed in Fig. 8. Since a bigger rectangular black area generally denotes overlapping/touching portion, all such bigger black areas are deleted from the smoothed image. The resultant image is shown in Fig. 9 and the deleted portions are marked by circles in this figure.

After deletion of the big black areas, the remaining black rectangular areas are dilated [19] to connect the black rectangular areas with the next stripes to make each line as a single component. The dilation operation is performed using a structuring element of length $4 \times SW$ and a width of one. Here SW is the width of the stripe defined in Section 2.1. The length of the structural element for dilation is experimentally selected based on analyzing 100 images from the three different handwritten databases (35 images from

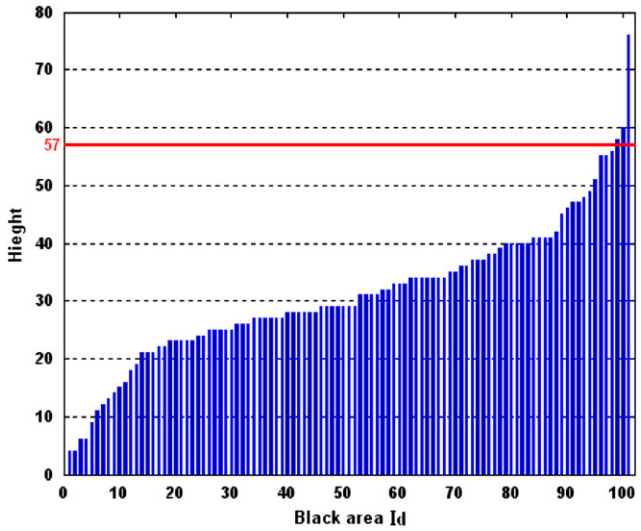

Fig. 8. Bar chart of the heights of black areas computed from Fig. 9. Here the threshold is considered as twice the average height of a black area and its value is 57.

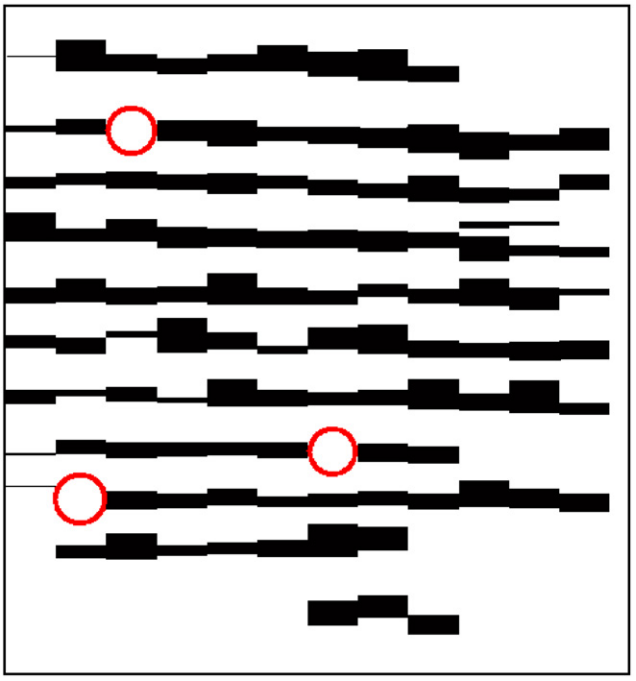

Fig. 9. Text-page after removing black color rectangles with height greater than a threshold (deleted portions are marked by a circle).

Ref. [9], 35 images from Ref. [13] and 30 images from our dataset). Using different structural elements of lengths $1 \times SW$, $2 \times SW$, $3 \times SW$, $4 \times SW$, $5 \times SW$, $6 \times SW$ and $7 \times SW$ we obtained segmentation accuracies of 68.34%, 89.46%, 93.08%, 96.57%, 95.24%, 95.28% and 85.20%, respectively. Therefore, we used structural element $4 \times SW$ for the testing of our proposed scheme. The result of the dilation operation on the image given in Fig. 9 is shown in Fig. 10.

2.3. Construction of separating lines

In text-line segmentation, each separator should pass through the background portions between two consecutive text-lines. Therefore, the complement of the dilated image obtained from the previous stage is used to get the background information. The complementary image is then thinned by applying the algorithm proposed in [20] to obtain some candidate line separators. The candidate line separators indicate the path information for final text-line segmentation. The result of the thinning operation is shown in Fig. 11.

The candidate line separators may have some junction points. In some cases, the candidate line separators may be joined at the junction point by a very small vertical line (in Fig. 11 such joining points are marked by dotted circles). For our segmentation process, we need to detach such candidate line separators by eliminating the small vertical lines or removing junction/intersection points. The Sobel filter with the following 3×3 mask [21] is employed for this purpose. This filter emphasizes horizontal edges (neglects vertical edges) using the smoothing effect by approximating a vertical gradient.

$$H = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

As a result of the filtering process, all the pixels belonging to vertical lines are eliminated. Therefore, the candidate line separators that are connected with other lines are subsequently disjoint. Then, the candidate line separators with a length less than the average length of candidate line separators are also eliminated

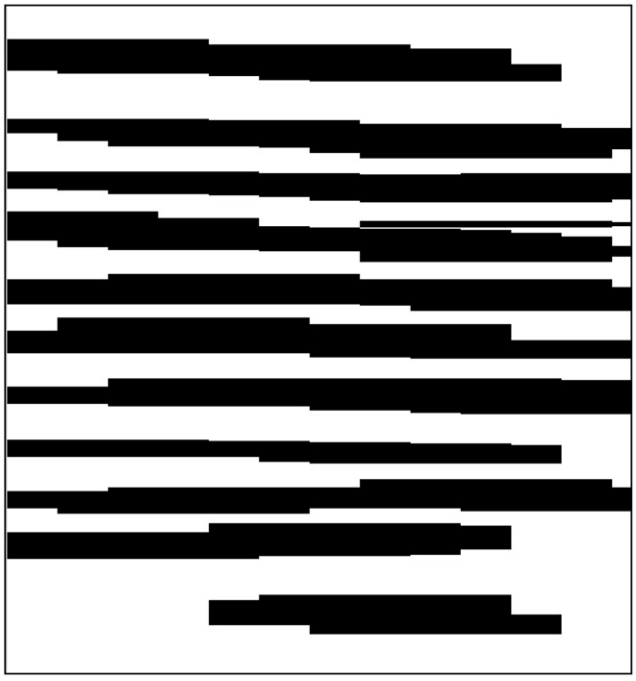

Fig. 10. Text-page after dilation operation.

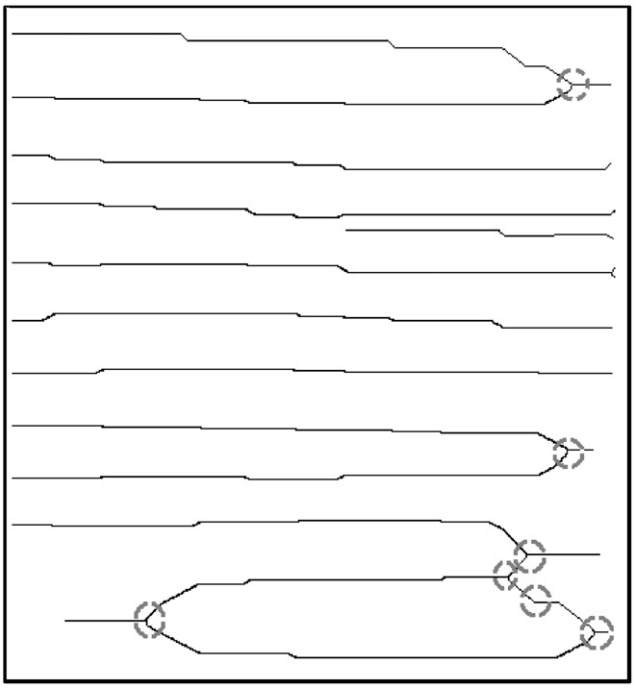

Fig. 11. Result after applying the thinning algorithm on background part of Fig. 10.

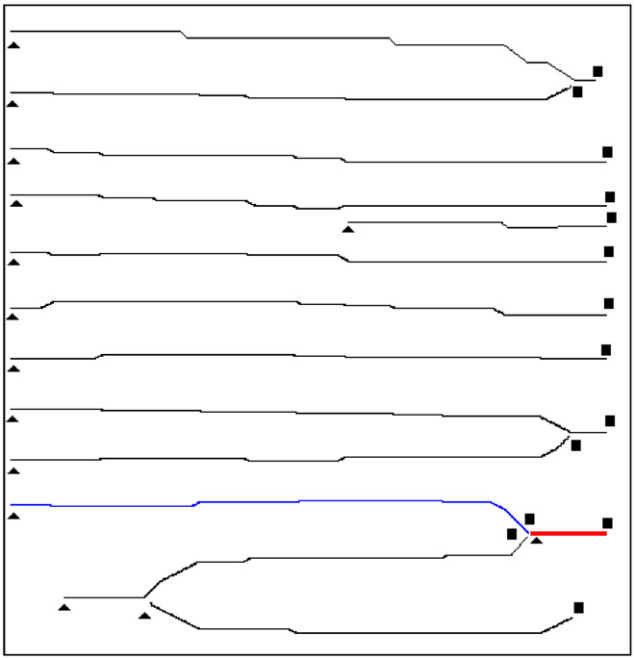

Fig. 12. Result after trimming operations (start and end points of each candidate line separator are indicated by triangles and rectangles, respectively. The red line is kept here for illustration). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

(e.g. red line in Fig. 12). The number of pixels in each line specifies the length of the line. The candidate line separators are also trimmed by removing only the diagonal parts that appear at the beginning or end of the candidate line separators. The result obtained after applying the filtering and trimming operations on the image given in Fig. 11 is shown in Fig. 12. The red line in Fig. 12 is kept for illustration of the next step.

In order to segment two consecutive text-lines of a document image, two or more candidate line separators should be joined to

get a complete separating line. To do that, directions of all the extracted candidate line separators are considered from left to right; starting and end points information of all the candidate separators are noted. Fig. 12 illustrates where start and end points of each candidate line separator are marked by triangles and rectangles, respectively. Since the direction of candidate line separators is considered from left to right, to construct a complete separating line it is necessary to join the end point of a candidate line separator to the starting point of the nearest candidate line separator located in the right side of that candidate separator. Analyzing the vertical distance between the end points of each candidate line separator (*CL*) with the starting points of all the other candidate line separators, whose starting points are located on the right side of the end point of the *CL*, the nearest candidate line separator (that has minimum vertical distance) is noted. If such minimum vertical distances are less than the average height of white areas in the image, then the *CL* is connected to its nearest candidate line separator [in Fig. 12 the blue candidate line separator (thin line) is connected to the red candidate line separator (thick line)]. If no such nearest candidate line separator is obtained to join the candidate line separator (*CL*), and if the length of the *CL* is less than the width of the input text image as well as the distances from both the sides (left/right) of the *CL* to the image boundaries (left/right) are less than half of the image width, then the *CL* is extended from both sides until it reaches both the left and right boundaries of the image. When the *CL* reaches the left and right boundaries of the image, the construction of the separating line is completed. At the end, the first separating line from the top (no text-line before it) and the candidate line separators that could not be joined to any other candidate separator and have a length less than the image width are eliminated. The resultant separating lines obtained from the input image of Fig. 2 are shown in Fig. 13.

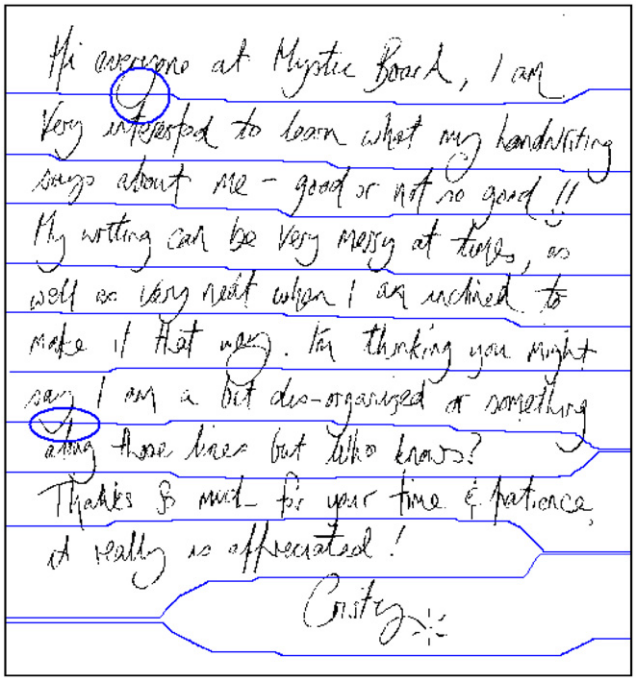

Fig. 13. Problems of touching and overlapping components, which are indicated with circles and ellipses, respectively.

2.4. Resolving the problems of overlapping and touching components

The text-line segmentation task cannot be completed without considering overlapping and touching cases (see Fig. 13, where some touching and overlapping cases are shown). If a separating line (say *SL*) does not pass through a black pixel of any text component then neither touching nor overlapping occurs between the two text-lines situated above and below the *SL*. Otherwise a problem of touching/overlapping occurs. In order to check whether overlap or touching has occurred, the height of the component having intersection with the *SL* is examined. If the height of the component is greater than T_h then the component is judged as a touching component. Otherwise, it is considered as an overlapping component. In the case of touching text-lines, at least two components from two consecutive text-lines touch and generate a single component. Thus, the height of a touching component is usually greater than twice the average height of the components present in the document image. Hence, T_h is defined as twice the average component height.

To take care of the problem of overlapping, the contour points of the component intersecting the *SL* are traced. The intersection point of the *SL* and the component is considered as a starting point for contour tracing. The direction of contour tracing (upwards or downwards) is selected based on the number of contour pixels that lie in the upper or lower parts with respect to the position of the starting point. If the number of contour pixels in the upper side of the starting point is smaller (greater) than the number of contour pixels in the lower side, then upwards (downwards) tracing is performed. The upwards or downwards traversal continues until it comes back to the same row of the starting point after reaching the top (bottom) point of the component so that the overlapped portion can be included in its text-line accordingly. Thus, if an overlap occurs, the separating line follows the contour of the overlapping component and the overlapping portion can be included in its respective text-line correctly. An overlapping situation and its step-by-step solution are demonstrated in Fig. 14.

For touching component segmentation, we first fix a touching zone of height T_1 [(statistical mode of heights of white spaces

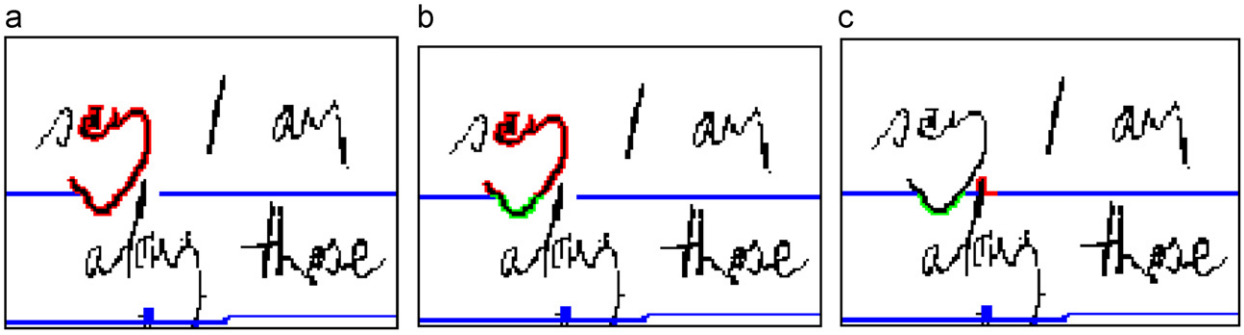

Fig. 14. Illustration of overlapping component assignment: (a) Finding the contour points of an overlapping component, (b) Dividing the contour points into two parts (upper and lower) based on the position of the intersection point and counting the number of contour points in both parts and (c) Assignment of overlapping component.

between two consecutive lines in the text image)/2] for a touching component. Touching generally occurs in this zone. A touching zone is shown in Fig. 15(a) and it is marked in gray. Based on a statistical study, it is observed that when two components are touching, in most cases, the position with minimum stroke width within the touching zone is the touching point of two components. In our statistical study, we have randomly chosen 20 samples of text-pages from the dataset used in [13]. The number of touching components in these sample text-pages was 68. Out of this, 56 touching components (82.4%) were correctly segmented by crossing the separating line through the position with minimum stroke width. Therefore, for touching segmentation, a separating line crosses a touching component through the position with minimum stroke width in the defined touching zone. A step-wise illustration of touching component segmentation is given in Fig. 15. Final segmentation results obtained from the original image in Fig. 2 by the proposed text-line segmentation algorithm are shown in Fig. 16(a).

It is worth mentioning that the proposed methodology can deal with touching of more than 2 consecutive lines. In case of touching with more than two consecutive text-lines, two or more searching zones for touching are obtained. Consequently, the touching component is gradually separated from the first text-line based on the respective searching zone and subsequently proceeds to the next consecutive lines.

Sometimes a component may be incorrectly considered as a touching component. In Persian and Arabic documents, this type of misclassification mostly occurs because of stretching of strokes and some characters are written at a larger size than the others. For documents written in English and other western languages, this type of error occurs sometimes because of a continuous writing style where two characters (one with an ascender and the other with a descender (such as "ℊ" and "ℎ")) touch each other in the same line. In this case, since the height of the connected component "ℊℎ" is greater than the threshold, the component "ℊℎ" is considered as a touching component. This type of error is mostly removed in our scheme using the following information. After segmenting this type of component into two parts, one of the parts is always a small component compared to the other one. Moreover, the position of this small component is far (defined below) from the adjacent

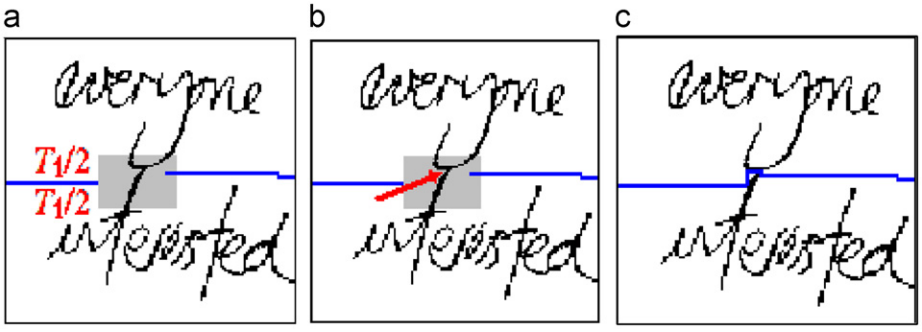

Fig. 15. Step-by-step illustration of touching component segmentation: (a) Fixing the searching zone based on threshold T1 (searching zone is marked in gray), (b) Finding the position of minimum width in the search zone (minimum width portion is marked by an arrow) and (c) Passing of separating line through the position with minimum width for touching segmentation.

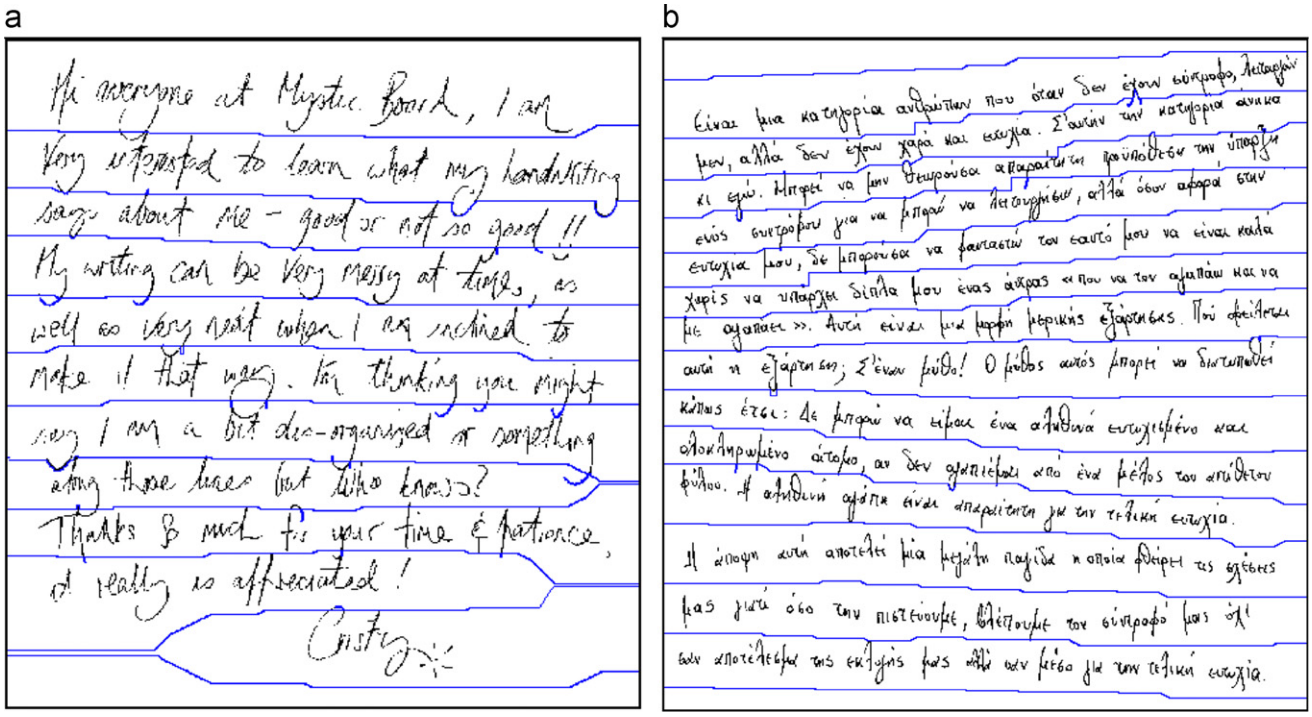

Fig. 16. Results of the proposed text-line segmentation algorithm (a) text-lines of the documents are mostly horizontal and (b) text lines have different skew orientations.

text-line. Hence, this small component cannot be included in the adjacent text-line. Therefore, the segmentation of this component into two parts is not considered and the overlapping process is performed to resolve it. Here a part with height smaller than half of the average component height is considered as "small" and "far" is defined as follows. Let a touching component be segmented into two parts and d_1 be the distance between the *CG*s of these two parts. Also let d_2 be the distance between the *CG* of the smaller component and the *CG* of its nearest component obtained from the text-line on which smaller component currently belongs to. If d_2 is greater than d_1, then we call the position of smaller component is "far" from adjacent text-line.

3. Experimental results and discussions

3.1. Dataset

The proposed text-line segmentation method is tested on three different datasets: (a) first, the proposed method is employed on 152 handwritten English, Greek, French, and German text documents containing 3382 lines [13]. The documents were collected from the historical archive of the University of Athens, from the Dallas library, and from the Handwriting Segmentation Competition of ICDAR2007 [24]. None of the test samples contain any non-text components. Moreover, each text sample contains several touching/overlapping components. Then we tested the proposed method with (b) 200 handwritten document images (4034 text-lines in total) used in the ICDAR2009 handwriting segmentation contest [9]. To demonstrate the applicability of the present work with text-pages written in other languages, we have further tested our scheme with (c) another dataset containing 78 handwritten text-pages (1044 text-lines in total) of Persian, Oriya and Bangla scripts written by different individuals and scanned in gray level with a resolution of 300 DPI.

3.2. Performance evaluation and experimental results

In the literature, most of authors have manually evaluated their text-line segmentation techniques [3,11,14]. Manual evaluation of the segmentation result is a very tedious and time-consuming task [13]. To evaluate the present scheme automatically, a similar evaluation strategy that was used in the ICDAR2009 handwriting segmentation contest [9] is also utilized here. The method of evaluating the performance generates two evaluation factors (detection rate (*Det*) and recognition accuracy (*Rec*)) based on counting the number of matches between the regions segmented by the algorithm and the regions in the ground truth [13,25]. The *Det and* the *Rec* refer to recall and precision, respectively, and the combination of the *Det* and the *Rec* is the *TLDM*. These measures [25] are calculated as follows:

$$Det = w_1 \frac{one2one}{N} + w_2 \frac{g\_one2many}{N} + w_3 \frac{g\_many2one}{N} \quad (4)$$

$$Rec = w_4 \frac{one2one}{M} + w_5 \frac{d\_one2many}{M} + w_6 \frac{d\_many2one}{M} \quad (5)$$

$$TLDM = \frac{2 \times Det \times Rec}{Det + Rec} \quad (6)$$

where *one2one* is a complete match between a pair if the matching score for this pair is greater than or equal to an acceptance threshold. A *g_one2many* match is a ground truth text-line, which partly matches two or more detected text-lines. A *g_many2one* match stands for two or more ground truth text-lines, which match one detected text-line to some extent. A *d_one2many* match is a detected text-line, which partially matches two or more ground truth text-lines. Finally, a *d_many2one* match is defined as two or more detected text-lines, which partially match one text-line in the ground truth. Parameter N is defined as the number of ground truth text-lines in each text-page, M is the number of separating text-lines obtained from the proposed scheme and W_1, W_2, W_3, W_4, W_5 and W_6 are pre-determined weights. While testing the proposed algorithm with 152 handwritten text-pages [13], the average correct text-line detection rate of 95.70% and a correct recognition accuracy of 95.17% are obtained when we follow the same parameters in [13]. In the second experiment, the proposed algorithm is tested with 200 handwritten text-pages from the ICDAR2009 segmentation contest [9] and 98.35% average correct text-line detection as well as 98.76% correct recognition accuracy were achieved.

Further, the proposed algorithm is tested with 78 unconstrained handwritten text-pages (1044 text-lines in total) of different languages such as Persian, Oriya and Bangla. We obtained 95.87% correct text-line detection with 94.34% correct recognition accuracy. The experimental results on different datasets and the values of all the parameters are displayed in Table 1. From the experimental results, it can be noted that the proposed algorithm is capable of segmenting handwritten text-pages with different contents, languages, and skews. Furthermore, we observed that text-line segmentation for unconstrained handwritten Persian, Oriya and Bangla text documents presents a more challenging and crucial problem because of the special characteristics of these scripts. To judge the performance of our scheme on different languages, some results of the proposed algorithm with different handwritten script samples are shown in Figs. 16(b) and 17(a–f).

It is worth mentioning that the proposed technique works well with the documents having a skew between -10° and $+10^\circ$. We noted that when the skew is beyond -20° and $+20^\circ$ our method gives incorrect segmentation results. For documents having such higher degree of skew, it is better to use skew information of the documents to achieve higher segmentation results. From the experiment, we also observed that our method might not work well when documents contain complex non-uniform background and it was mainly due to improper binarization.

From the perspective of analysis of computing time, the overall time complexity of the proposed method is $O(n^2)$ where n^2 is the total number of pixels in the input image.

Table 1
Experimental results of the proposed technique on different datasets.

Dataset	Parameters and results									
	M	one2 one	g_one2many	g_many2 one	d_one2 many	d_many2one	Values of w_i	Det. rate (%)	Rec. Acc. (%)	TLDM (%)
152 text-pages [13]	3403	3216	20	70	33	38	$w_2=w_3=w_5=w_6=0.25$, and $w_1=w_4=1$	95.70	95.17	95.43
200 text-pages [9]	4017	3967	–	–	–	–	$w_2=w_3=w_5=w_6=0$, and $w_1=w_4=1$	98.35	98.76	98.55
78 Persian, Oriya and Bangla text-pages	1044	985	–	–	–	–	$w_2=w_3=w_5=w_6=0$, and $w_1=w_4=1$	95.87	94.34	95.10

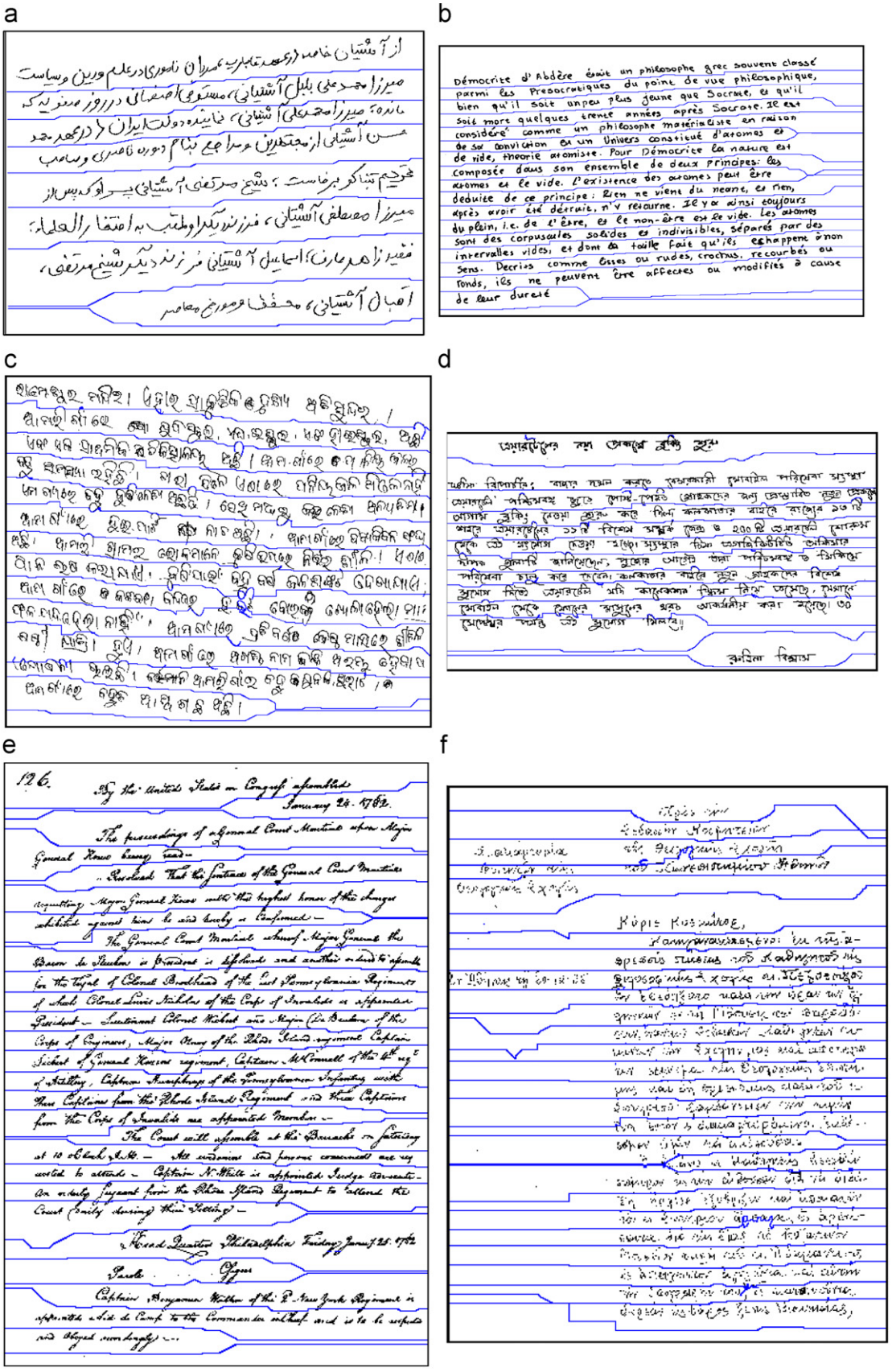

Fig. 17. (a) Result of the proposed text-line segmentation algorithm on a Persian text-page, (b) Result of the proposed text-line segmentation algorithm on a text-page from the dataset used in [9], (c) Result of the proposed text-line segmentation algorithm on an Oryia text-page, (d) Result of the proposed text-line segmentation algorithm on a Bangla text-page, (e) Result of the proposed text-line segmentation algorithm on a text-page of the dataset used in [13] and (f) Result of the proposed text-line segmentation algorithm on a text-page of the dataset used in [13].

3.3. Analysis of errors

Some errors of the proposed algorithm obtained during testing are shown in Fig. 18(a)–(c). The error shown in Fig. 18(a) occurred due to the incorrect segmentation of a touching component. Here the position of the minimum stroke width for crossing the component is outside the searching zone, which is the main reason for the occurrence of this kind of error. In Fig. 18(b), the proposed scheme has segmented the image into 5 lines. There is an over-segmentation (in the 3rd line) which is due to (i) a substantial horizontal gap between the 2nd and 3rd word of the 3rd line, and (ii) less horizontal overlap between them. Because of such behavior in the text-line, at the time of the painting process there appeared to be disconnectivities within a text-line and it was not possible to get a single component to consider it as a single text-line, hence oversegmentation resulted. Fig. 18(b) is one of the rare cases that occurred during our experiments and we believe that it is a step-like text-line (not a skewed text-line). Based on visual observation without considering semantic information, we can consider that the proposed algorithm has given the correct result on the image. Moreover, by looking at the image shown in Fig. 18(d) we may consider it as one text-line. However, in the respective ground truth data it has been considered as two separate text-lines. As with the 3rd line of Fig. 18(b) our method also segmented the text-line shown in Fig. 18(d) into two text-lines and in this case the segmentation result was correct. It must also be noted that if there is overlap between the words in a text-line then a substantial horizontal gap between the words does not affect the performance of the proposed text-line segmentation method.

Sometimes a component was incorrectly treated as a touching component (see Fig. 18(c)). This type of error occurred in a document that has a few components with substantial sizes (heights) compared to the rest of the components of the document. In this case, since the heights of some components are greater than the threshold, they are considered as touching components.

3.4. Comparative analysis

To perform an evaluation with the state of the art results in the literature, we compare the results of the present work with the results of some available algorithms reported in [13]. The same dataset (152 text-pages) of [13] is used here for the comparison of results. A detailed comparison of different methods is shown in Table 2. From Table 2, it is evident that the results reported in the proposed technique outperformed the results of all other algorithms.

Furthermore, the results of the proposed technique on the 200 text-images of the ICDAR2009 handwriting contest are compared with the results reported in the handwriting segmentation contest [9] of ICDAR2009. Table 3 shows a comparison of the results. From Table 3, it can be noted that the proposed technique has outperformed the techniques based on graph analysis (PortoUniv), Connected Component Labeling (Jadavpur Univ), spanning tree analysis (CASIA-MSTSeg), labeling technique (CMM) and smearing

Table 2
Comparison of our results with different algorithms used in [13] and tested on 152 text-pages.

Methods	Percentages		
	Detection rate (%)	Recognition Accuracy (%)	TLDM (%)
Block-based Hough transform [13]	95.80	93.80	94.80
Fuzzy run-length [14]	83.00	78.50	80.70
Hough approach [7]	86.60	74.20	80.00
Projection profiles [2]	70.20	73.70	71.90
Proposed method	**95.70**	**95.17**	**95.43**

Table 3
Comparison of results of different algorithms reported in ICDAR09 [9] with the proposed method.

Methods	Parameters and results				
	M	one2one	Detection rate (%)	Recognition accuracy (%)	TLDM (%)
CUBS	4036	4016	99.55	99.50	99.53
ILSPLWSeg-09	4043	4000	99.16	98.94	99.05
PAIS	4031	3973	98.49	98.56	98.52
CMM	4044	3975	98.54	98.29	98.42
CASIAMSTSeg	4049	3867	95.86	95.51	95.68
PortoUniv	4028	3811	94.47	94.61	94.54
PPSL	4084	3792	94.00	92.85	93.42
LRDE	4023	3901	96.70	88.20	92.25
Jadavpur Univ	4075	3541	87.78	86.90	87.34
ETS	4033	3496	86.66	86.68	86.67
AegeanUniv	4054	3130	77.59	77.21	77.40
REGIM	4563	1629	40.38	35.70	37.90
Proposed method	**4017**	**3967**	**98.35**	**98.76**	**98.55**

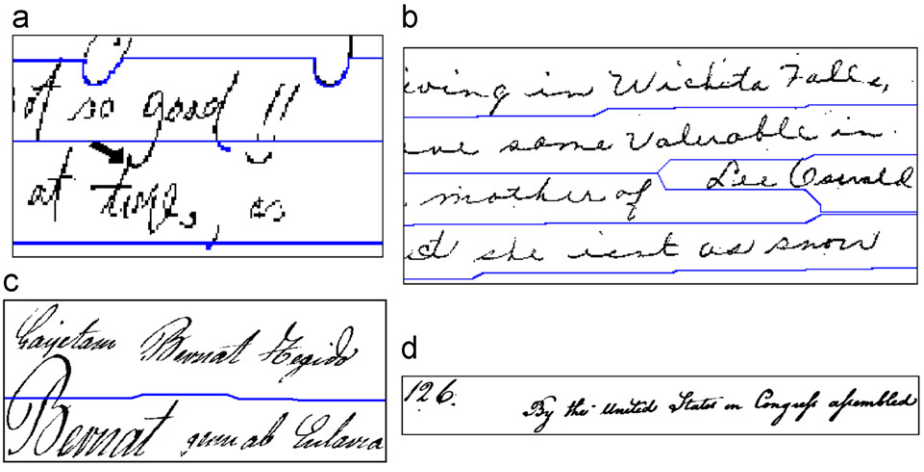

Fig. 18. (a) Error because of improper segmentation of a touching component in the text-line whereby the correct segmentation point is indicated with an arrow, (b) Error due to step-like text line of the 3rd line, (c) Error because of the unusual size of a component with respect to the components of the whole image and (d) Example of a text line considered as two text lines in the ground truth data.

method (ETS) as well as all the piece-wise segmentation based techniques (AegeanUniv, PAIS, LRDE, PPSL, REGIM) except the ILSPLWSeg-09 method. The technique based on directional run-length analysis (CUBS) has also shown the best results.

4. Conclusion and future work

In this paper, a novel script independent algorithm for handwritten text-line segmentation is presented. The generality of the proposed algorithm is demonstrated by testing the algorithm with different handwritten text-pages written in different languages. The proposed algorithm is evaluated with two datasets and it shows better segmentation accuracies compared with the results of most of the recent methods available in the literature. Moreover, we have tested the present algorithm with different text-pages written in Persian (Farsi), Oriya and Bangla scripts. While testing the algorithm, some errors have occurred due to (i) incorrect segmentation of touching/overlapping components and (ii) lack of overlap between two consecutive words in a text-line.

In future work, the authors plan to develop some post-processing techniques for efficient handling of the touching and over-segmentation problems to obtain higher accuracies from the proposed segmentation scheme.

Acknowledgements

The authors would like to thank the anonymous reviewers of the paper for their valuable comments and constructive suggestions, which considerably improved the quality of the paper. We would also like to thank Dr. Basilis Gatos of the Computational Intelligence Laboratory, National Center for Scientific Research, Greece for providing the handwritten datasets as well as evaluation software used in our experiments, and Dr. Michael Blumenstein of Griffith University, Australia for his help in this work.

References

[1] M.R. Hashemi, O. Fatemi, R. Safavi, Persian Cursive Script Recognition, in: Proceedings of the Third International Conference on Document Analysis and Recognition, 1995, pp. 869–873.
[2] R. Manmatha, J.L. Rothfeder, A scale space approach for automatically segmenting words from historical handwritten documents, IEEE Transactions on Pattern Analysis and Machine Intelligent 27 (8) (2005) 1212–1225.
[3] U. Pal, S. Datta, Segmentation of Bangla unconstrained handwritten text, in: Proceedings of the Seventh International Conference on Document Analysis and Recognition, 2003, pp. 1128–1132.
[4] A. Zahour, B. Taconet, P. Mercy, S. Ramdane, Arabic hand-written text-line extraction, in: Proceedings of the Sixth International Conference on Document Analysis and Recognition, 2001, pp. 281–285.
[5] A.Zahour, L. Likforman-Sulem, W. Boussalaa, B. Taconet, Text-line segmentation of historical Arabic documents, in: Proceedings of Ninth International Conference on Document Analysis and Recognition, 2007, pp. 138–142.
[6] M. Arivazhagan, H. Srinivasan, S.N. Srihari, A statistical approach to handwritten line segmentation, in: Proceedings of SPIE Document Recognition and Retrieval XIV, 2007, p. 6500T-1–11.
[7] V. Papavassiliou, T. Stafylakis, V. Katsouros, G. Carayannis, Handwritten document image segmentation into text lines and words, Pattern Recognition 43 (2010) 369–377.
[8] Y. Li, Y. Zheng, D. Doermann, S. Jaeger, Script-Independent Text Line, Segmentation in freestyle handwritten documents, IEEE Transaction on Pattern Analysis and Machine Intelligence 30 (8) (2008) 1313–1329.
[9] B. Gatos, N. Stamatopoulos, G. Louloudis, ICDAR2009 Handwriting Segmentation Contest, in: Proceedings of 10th International Conference on Document Analysis and Recognition, 2009, pp. 1393–1397.
[10] L. Likforman-Sulem, A. Zahour, B. Taconet, Text-line segmentation of historical documents: a survey, International Journal on Document Analysis and Recognition 9 (2) (2007) 123–138.
[11] L. Likforman-Sulem, A. Hanimyan, C. Faure, A Hough based algorithm for extracting text-lines in handwritten documents, in: Proceedings of the Third International Conference on Document Analysis and Recognition, 1995, pp. 774–777.
[12] G. Louloudis, B. Gatos, I. Pratikakis, K. Halatsis, A. Block-Based Hough, Transform mapping for text-line detection in handwritten documents, in: Proceedings of 10th International Workshop on Frontiers in Handwriting Recognition, 2006, pp. 515–520.
[13] G. Louloudis, B. Gatos, I. Pratikakis, C. Halatsis, Text-line detection in handwritten documents, Pattern Recognition 41 (2008) 3758–3772.
[14] Z. Shi, V. Govindaraju, Line separation for complex document images using fuzzy runlength, in: Proceedings of First International Workshop on Document Image Analysis for Libraries, 2004, pp. 306.
[15] P.P. Roy, U. Pal, J. Lladós, Morphology based handwritten line segmentation using foreground and background information, in: Proceedings of International Conference on Frontiers in Handwriting Recognition, 2008, pp. 241–246.
[16] S.S. Bukhari, F. Shafait, T.M. Breuel, Script-independent handwritten textlines segmentation using active contours, in: Proceedings of 10th International Conference on Document Analysis and Recognition, 2009, pp. 446–450.
[17] F. Yin, C.L. Liu, A variational bayes method for handwritten text line segmentation, in: Proceedings of 10th International Conference on Document Analysis and Recognition, 2009, pp. 436–440.
[18] S. Tsuruoka, Y. Adachi, T. Yoshikawa, The Segmentation of a text-line for a handwritten unconstrained document using thinning algorithm, in: Proceedings of Seventh International Workshop on Frontiers in Handwriting Recognition, 2000, pp. 505–510.
[19] Van den R. Boomgard, R. van Balen, Methods for fast morphological image transforms using bitmapped images, Computer Vision, Graphics, and Image Processing: Graphical Models and Image Processing 54 (3) (1992) 254–258.
[20] L. Lam, L. Seong-Whan, Y.Suen Ching, Thinning methodologies—a comprehensive survey, IEEE Transactions on Pattern Analysis and Machine Intelligence 14 (9) (1992) 879.
[21] Rafael C. Gonzalez, Richard E. Woods, Digital Image Processing, second edition, Prentice Hall, India, 2002.
[22] B. Su, S. Lu, C.L. Tan, A self-training learning document binarization framework, in: Proceedings of 20th International Conference on Pattern Recognition, 2010, pp. 3187–3190.
[23] N. Otsu, A threshold selection method from gray-level histograms, IEEE Transaction on System, Man. and Cybernetics 9 (1) (1979) 62–69.
[24] B. Gatos, A. Antonacopoulos, N. Stamatopoulos, ICDAR2007 handwriting segmentation contest, in: Proceedings of Ninth International Conference on Document Analysis and Recognition, 2007, pp. 1284–1288.
[25] I. Phillips, A. Chhabra, Empirical performance evaluation of graphics recognition systems, IEEE Transaction on Pattern Analysis and Machine Intelligence 21 (9) (1999) 849–870.

Alireza Alaei received his B.E. in Computer Software Engineering and M.Sc. in Computer Science in 1995 and 2007 from Shahid Beheshti University, Tehran, Iran and University of Mysore, Mysore, India, respectively. He currently is a research scholar at the Department of Studies in Computer Science, University of Mysore, Mysore, India for pursuing his Ph.D. in Computer science on the area of handwritten document image analysis and recognition. He has published more than 10 research papers in international conferences. His area of interest is document image processing, pattern recognition, machine learning and image processing. He is a member of IEEE.

Umapada Pal received his Ph.D. from Indian Statistical Institute (ISI). He did his Post Doctoral research at INRIA, France. During July 1997–January 1998 he visited GSF-Forschungszentrum fur Umwelt und Gesundheit GmbH, Germany as a guest scientist. From January 1997, he is a faculty member of the Computer Vision and Pattern Recognition Unit, ISI, Kolkata. He has published 160 research papers in various international journals, conference proceedings, and edited volumes. He received student best paper award from Chennai Chapter of Computer Society of India and a merit certificate from Indian Science Congress Association in 1995 and 1996, respectively. Dr. Pal achieved 'ICDAR Outstanding Young Researcher Award' from TC-10 and TC-11 committees of IAPR in 2003. He has been serving as a guest editor, co-editor, program chair, and program committee member of many international journals and conferences. He is a life member of Indian unit of IAPR and a senior life member of Computer Society of India.

P. Nagabhushan (B.E.—1980, M.Tech.—1983, Ph.D.—1989) is a professor at the Department of Studies in Computer Science and was Director—Planning Monitoring and Evaluation Board at the University of Mysore, India. He is an active researcher in the areas pertaining to Pattern Recognition, Document Image Processing, Symbolic Data Analysis and Data Mining. Till now he has successfully supervised 18 Ph.D. candidates. He has over 400 publications in journals and conferences of International repute. He has chaired several international conferences. He is a visiting professor to USA, Japan and France. He is a fellow of Institution of Engineers and Institution of Telecommunication and Electronics Engineers, India.