

# Script-Independent Text Line Segmentation in Freestyle Handwritten Documents

Yi Li, *Student Member, IEEE*, Yefeng Zheng, *Member, IEEE*,  
 David Doermann, *Member, IEEE*, and Stefan Jaeger

**Abstract**—Text line segmentation in freestyle handwritten documents remains an open document analysis problem. Curvilinear text lines and small gaps between neighboring text lines present a challenge to algorithms developed for machine-printed or hand-printed documents. In this paper, we propose a novel approach based on density estimation and a state-of-the-art image segmentation technique, the level set method. From an input document image, we estimate a probability map where each element represents the probability of the underlying pixel belonging to a text line. The level set method is then exploited to determine the boundary of neighboring text lines by evolving an initial estimate. Unlike connected component-based methods ([1] and [2], for example), the proposed algorithm does not use any script-specific knowledge. Extensive quantitative experiments on freestyle handwritten documents with diverse scripts such as Arabic, Chinese, Korean, and Hindi demonstrate that our algorithm consistently outperforms previous methods [1], [2], [3]. Further experiments show that the proposed algorithm is robust to scale change, rotation, and noise.

**Index Terms**—Handwritten text line segmentation, document image analysis, density estimation, level set methods.

## 1 INTRODUCTION

TEXT line segmentation is one of the major components of document image analysis. It provides crucial information for skew correction, zone segmentation, and character recognition. Although text line segmentation for machine-printed or hand-printed documents is usually seen as a solved problem [4], freestyle handwritten text lines still present a significant challenge. In particular:

1. Handwritten text lines are curvilinear. Linear or piecewise linear approximation is not accurate in general.
2. Neighboring handwritten text lines may be close or touch each other. No well-defined baselines exist for most handwritten documents.

Fig. 1 shows the segmentation results of the Docstrum algorithm [1], a well-known layout analysis algorithm, on a machine-printed document and a freestyle handwritten document. As we can see, although the algorithm works reasonably well on the machine-printed document, it is still a significant challenge to segment text lines in handwritten documents. Bounding boxes are widely used in machine-printed document analysis to represent the segmentation results of zones, text lines, words, and characters. However,

it is not trivial to use bounding boxes to label overlapping curvilinear text lines, as shown in Fig. 1a. A closed curve (or a polygon as an approximation) more appropriately represents the boundary of a text line. Previous text line segmentation methods, such as connected component-based methods, directly work on the input image (generally, a binary image). For connected component analysis, each pixel is equally treated and a change of one pixel may result in a different result. If two neighboring text lines touch each other through even a single handwritten stroke, the segmentation algorithm fails. We deal with this problem by using a probabilistic approach. The distribution of black pixels in a document is not uniform. At the center of a text line, a pixel is more likely to be a black pixel than at a text line gap. Therefore, we estimate a probability map where each pixel represents the probability that it belongs to a text line. As shown in Fig. 2b, the text line structure is enhanced. Based on the probability map, the boundaries of text lines can be further determined using region growth techniques. To the best of our knowledge, this is the first paper to study the text line segmentation problem from the density estimation perspective. In this paper, we make the following contributions:

1. Unlike previous methods, we estimate the probability density function (PDF) of 2D variables, which represents the text line distribution.
2. With the a priori knowledge that a text line is a horizontally elongated shape, we use a state-of-the-art region growth technique to determine the text line boundary based on the PDF.
3. Our approach combines the advantages of the bottom-up and top-down approaches, that is, the script independence of the projection-based methods and the capability of processing a complex layout of the connected-component-based methods.

• Y. Li and D. Doermann are with the Language and Media Processing Laboratory, Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742.  
*E-mail:* {liyi, doermann}@cfar.umd.edu.

• Y. Zheng is with Siemens Corporate Research, 755 College Road East, Princeton, NJ 08540. *E-mail:* yefeng.zheng@siemens.com.

• S. Jaeger is with the CAS-MPG Partner Institute for Computational Biology, Group of Systems Bioinformatics, 320 Yue Yang Road, Shanghai 200031, China. *E-mail:* jaeger@picb.ac.cn.

Manuscript received 7 Dec. 2006; revised 5 June 2007; accepted 7 Aug. 2007; published online 1 Oct. 2007.

Recommended for acceptance by L. O'Gorman.

For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMI-0855-1206. Digital Object Identifier no. 10.1109/TPAMI.2007.70792.

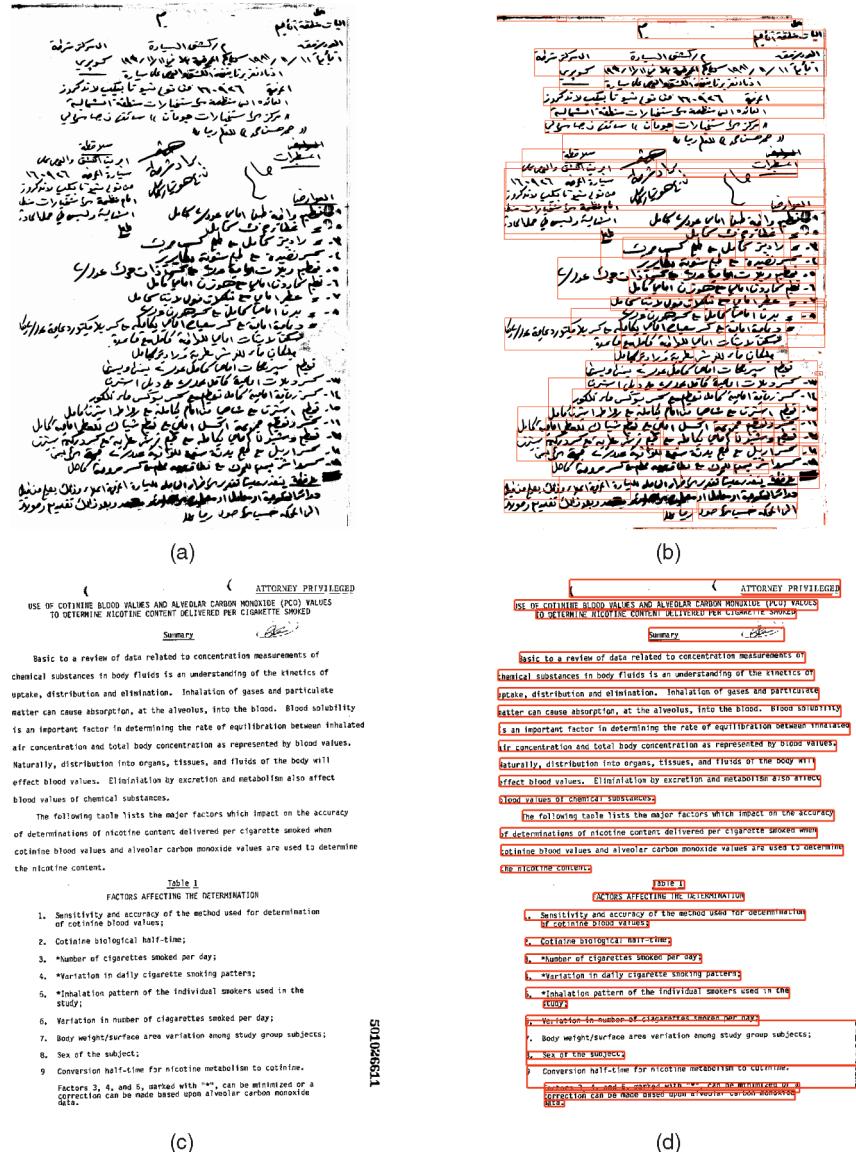


Fig. 1. (a) A handwritten document image. (b) Text line segmentation result of (a) using the Docstrum method [1] by using bounding boxes. (c) A machine-printed document image. (d) Text line segmentation result of (c) using the Docstrum method [1] by using bounding boxes.

Preliminary results of the proposed approach were published in our previous conference proceedings [5], [6]. In this paper, we discuss a novel view of our approach and present extensive quantitative comparison experiments on a large handwriting data set using approximately 10 scripts.

This paper is organized as follows: In Section 2, we review some related materials. Text line density estimation and text line boundary evolution using the level set method are described in Sections 3 and 4, respectively. Section 5 presents an extensive performance evaluation and quantitative comparison of the proposed method and the previous methods on a large handwriting data set. This paper ends with a summary and a brief discussion of our future work.

## 2 RELATED WORK

In this section, we review previous work on text line segmentation in machine-printed documents and some extensions for handwritten documents. The limitations of

the previous work on handwritten documents are also discussed.

### 2.1 Layout Hierarchy in Document Image

A document can be viewed as an organization of a hierarchical structure of pages, zones, text lines, words, and characters. Layout analysis attempts to extract this hierarchical structure from an image representation [7], [8]. To segment a document at a specified level, previous work on machine printed document analysis relies on the homogeneity inside a region and a significant gap between neighboring regions [9]. Two methodologies, bottom up and top down, were frequently used in previous work for layout analysis [10]. In the bottom-up methodology, connected components are grouped into lines, lines are grouped into zones, and so on. In the top-down methodology, a document page is first segmented into zones, a zone is further segmented into lines, and so on.

Although the techniques for machine printed documents are mature, it is much more challenging to analyze a

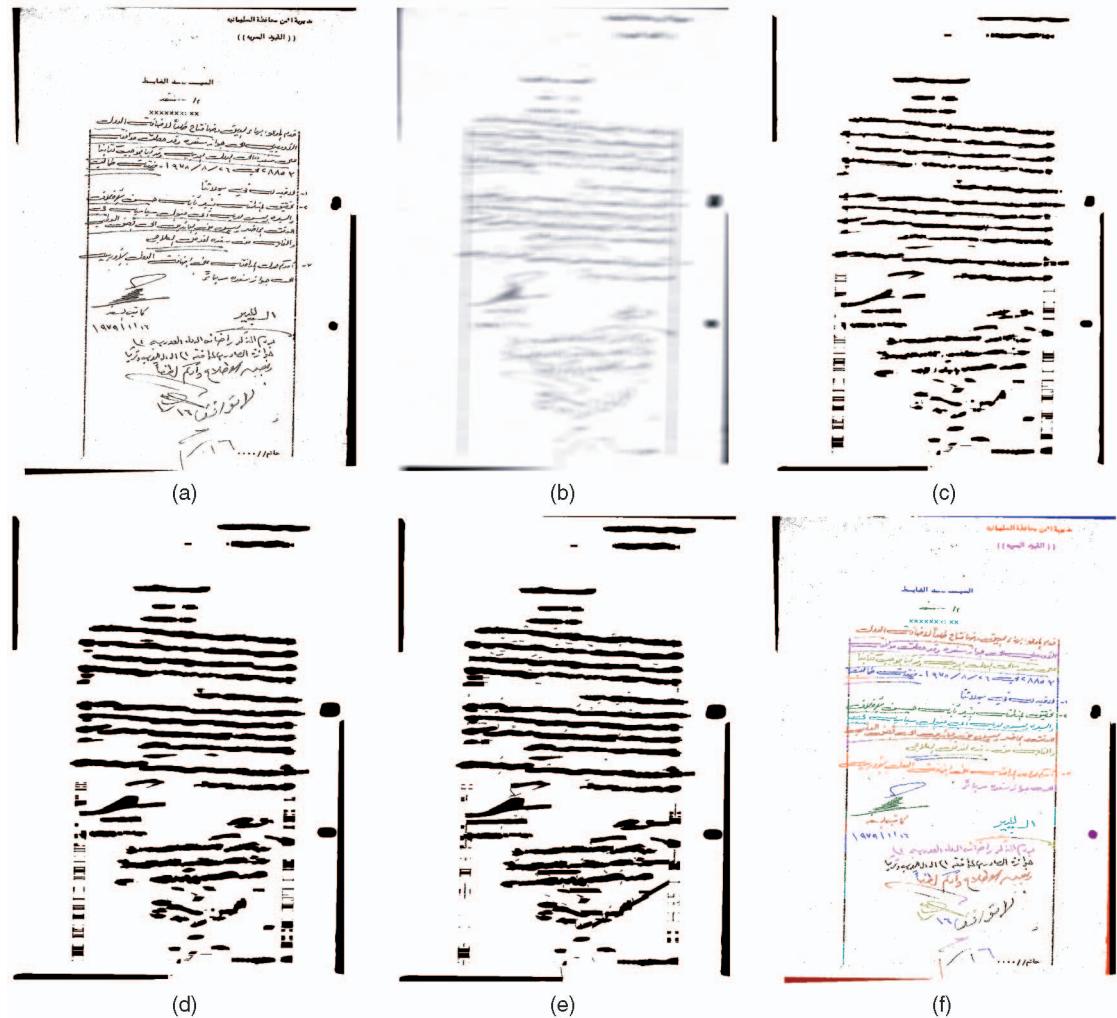


Fig. 2. Illustration of the proposed text line segmentation method. (a) A handwritten document. (b) Density estimation using anisotropic kernel. (c) Initial estimate of text lines. (d) Result after 10 iterations of evolution using the level set method. (e) After connecting a few broken text lines and grouping the isolated connected components (postprocessing). (f) Final text line segmentation result.

freestyle handwritten document due to several reasons, including:

1. Non-Manhattan layout at the page level. The irregular page layout cannot be handled by simple rules such as grouping based on geometric relationships of neighboring components.
2. Curvilinear text lines. Linear approximation and regression are not always accurate in general.
3. Connected words. Due to the connectivity between characters and words, it is not trivial to analyze, filter, and process the connected components.
4. Character size variation. Even on the same page, character size can be quite different. It causes problems in estimating the dominant character size, which is an important parameter for the bottom-up connected component-based text line segmentation algorithms [1].

## 2.2 Text Line Segmentation

Text line segmentation is important in layout analysis because it serves as the intermediary between page analysis and word segmentation. For example, if we obtain the text

lines, we can group them into zones based on the similarity of orientation and alignment or we can further segment text lines into words. Therefore, text lines are crucial for analyzing the hierarchy of document layouts.

As a special case of layout analysis, previous text line segmentation algorithms can roughly be categorized as bottom up or top down. In the bottom-up approach, connected component-based methods [2], [11] merge neighboring connected components by using rules based on the geometric relationship between neighboring blocks, such as distance, overlap, and size compatibility. Although connected component grouping techniques can process complex layouts, these methods are sensitive to topological changes in the connected components. For example, a handwritten stroke may cross two neighboring text lines, making two words in two text lines inseparable at the component level. Another disadvantage is that they may be script dependent. For example, spaces occur between neighboring words in English, but a Chinese text line is composed of a string of characters without word spaces. Therefore, it is difficult to derive a script-independent merging rule based on connected components.

Projection-based methods [3] may be one of the most successful top-down algorithms for machine printed documents. It can be used in many problems such as deskewening [12]. It assumes that the gap between two neighboring text lines is significant, and the projection of text lines is easily separable in the  $x$  or  $y$  direction. However, this method cannot be directly used in freestyle handwritten documents unless gaps between two neighboring text lines are significant or handwritten text lines are straight [13], [14]. The top-down approaches also have the disadvantage that they cannot process complex non-Manhattan layouts.

Text line segmentation approaches for machine printed documents are often based on the assumption of straight text lines. This assumption fails in a freestyle handwritten document, where text lines are often curvilinear. Some approaches, such as the projection-based methods, can be extended to deal with slightly curved text lines. Tripathy and Pal [15] proposed a method of dividing the image into multiple columns by using a projection-based method in each column and combining the results of adjacent columns into a longer text line. Generally, they achieved better results than naive methods, but the results of two adjacent columns may be ambiguous, so it is still difficult to generate good results. The performance of connected component-based methods on a handwritten document can also be improved by estimating the local orientation of the text line and using it to guide the merging of connected components [2]. Other researchers proposed different assumptions for their specific case [16] or for a specific script [17].

Rule lines often guide a person's writing [18]. Writers either write above rule lines (for example, in Chinese) or cross the rule lines (for example, in Hindi). If the writing follows the rule lines, text line segmentation may be easier since rule lines provide hints about the text line structure. However, if rule lines are ignored by the writer, they degrade the performance. For example, rule lines make connected component analysis difficult. Projection profile analysis also has problems in determining peaks and valleys given rule lines. In freestyle handwritten documents, most cases lie in the middle of these two extremes. Some text lines follow the rule lines well, while other text lines may cross multiple rule lines.

### 3 TEXT LINE DENSITY ESTIMATION

In this section, we describe the motivation for estimating a probability map from an input image and discuss the nonparametric method used in our approach. Without loss of generality, we assume that character pixels are black and are represented as "1"s in a binary image, while background pixels are white and "0"s. We also assume that the orientations of text lines are locally uniform and a skew correction method can normalize the text lines to orient horizontally. A relatively large variation in orientation, for example, up to  $\pm 10^\circ$ , is still allowed. However, zone segmentation [19] may be necessary to separate text lines with different orientations (for example, horizontal versus vertical) into different zones if they appear on the same document.

#### 3.1 Motivation

For decades, directly performing segmentation on a binary image has been accepted without criticism, although it

happens that some difficulties are inherently involved. No evidence exists that humans solely use traditional methods such as projection profile analysis to segment handwritten text lines. The computer algorithms, of course, do not necessarily mimic human behavior, but human behavior may give us some insights to explore for better algorithms. A certain pattern naturally occurs when human beings write text lines in documents. For example, the writing speed may remain almost constant within a paragraph, so distances between two adjacent text lines are usually similar. This motivates us to consider the PDF that generates this pattern. Instead of directly segmenting on a binary image, we convert it to a probability map, where each element represents the probability of this pixel belonging to a text line. For visualization, we rescaled the probability map to a gray-scale image (see Fig. 2b). This estimated probability map offers many working advantages. We can avoid calculating some geometric properties such as centroid and connected component size in the beginning because they are not accurate for handwritten documents. Another advantage is that we can adopt many state-of-the-art algorithms to analyze the two-dimensional density function. Finally, our method is not limited to binary images and can be used for gray-scale and color images.

The black pixels in an image can be regarded as two-dimensional random variables. They are generated by an unknown PDF. This PDF represents the distribution of text lines. Specifically, the PDF is continuous and has larger values in the text line area, while there are smaller values in the gap and marginal area. Our approach estimates the PDF from black pixels, which are the random variables. Peaks on the probability map represent text lines, while valleys are the boundary between neighboring text lines. In our approach, this probability map is analyzed using the level set method [20] to segment text lines.

Real problems often involve multimodal densities whose density functions are usually unknown. Therefore, nonparametric density estimation should be used. The Parzen window [21] is one of the nonparametric density estimation techniques. Isotropic kernels are frequently used if no domain-specific prior is given [22]. Without loss of generality, we assume the reading order of text lines to be left to right (or right to left) in handwriting, so the text line orientation is generally horizontal. Therefore, an anisotropic kernel is more appropriate to estimate the density. A simple anisotropic kernel can be a 2D Gaussian with different standard deviations in the  $x$  and  $y$  directions.

#### 3.2 Density Estimation Using an Anisotropic Kernel

Given an input binary image  $I(x, y)$ , for  $x = 1, \dots, M$  and  $y = 1, \dots, N$ , we want to estimate the probability map  $PDF_{est}(x, y)$  for  $1 \leq x \leq M$  and  $1 \leq y \leq N$  from the observation  $I(x, y)$ . Kernel-based nonparametric methods are widely used in statistics. Given a continuous Gaussian kernel  $\varphi$  in a 2D space, the kernel-based estimate is

$$PDF_{est}(x, y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \varphi(u, v) I(x - u, y - v) du dv. \quad (1)$$

For computational efficiency, the support region of the Gaussian kernel is often truncated in discrete implementation. If the standard deviations in the  $x$  and  $y$  directions are

$\sigma_x$  and  $\sigma_y$ , respectively, the window size is chosen as  $3\sigma_x$  and  $3\sigma_y$  to minimize the truncation error. The estimated probability map  $PDF_{est}(x, y)$  has high density values (dark, as shown in Fig. 2b) on the text lines but low density values (light) for the gaps.

Parzen window techniques guarantee that the result PDF is continuous and smooth if a Gaussian kernel is adopted, which is good for boundary selection. For example, we can use the gradient-based methods for our purpose, as presented in the next section. In addition to its theoretical advantage, this technique is also practically meaningful. As shown in Fig. 2b, the details that are irrelevant to the task of text line segmentation such as the script, content of the text, and word gaps have been blurred, while the text line structures are enhanced.

## 4 DETERMINING THE TEXT LINE BOUNDARY

Peaks on the estimated probability map represent text lines, while valleys are between neighboring text lines boundaries, as shown in Fig. 2b. We use a closed curve to represent the text line boundary. To obtain a unique segmentation, two adjacent text line regions should not overlap and their boundaries should not cross. Thresholding may give a reasonable segmentation result. However, the segmented text lines may be fragmented, as shown in Fig. 2c. This initial estimate of the text line boundary must evolve by considering the image information to obtain the final accurate segmentation results. Similar problems related to continuous regions and their boundaries have been studied in modern geometry and fluid mechanics theories. The continuous closed boundaries evolve to their desired location under the guidance of a partial differential equation (PDE). The level set method [20] is an effective algorithm in which the closed level set can automatically grow and merge. It has successfully been adopted in many fields, such as fluid mechanics, computer animation, and computer vision. The level set method naturally deals with topological changes and guarantees no overlapping in different regions, so it is well suited for our application. In this section, we adopt the level set method for the boundary evolution.

### 4.1 Level Set Method

For the basic idea of the level set method, the boundary evolves by its partial derivative and an external vector field. A specific formulation of the level set method can be written as follows [23]:

$$\frac{\partial f}{\partial t} + S_N \nabla f = b_k |\nabla f|, \quad (2)$$

where  $f$  is the implicit function and its zero level set represents a boundary,  $t$  denotes the time, and  $\frac{\partial f}{\partial t}$  represents the movement of the implicit function across time. The movement is controlled by  $b_k$ , which is usually the curvature of the boundary, and the scalar  $S_N$ , which is the movement speed of the boundary in the normal direction.

In the level set method, the closed boundary of a text line is represented by an implicit function  $f$ , which has negative values inside, positive values outside, and a zero value for the boundary. We then evolve the function and keep tracking the zero level set according to the PDE (2). Thus, the zero level set automatically grows, merges, and stops to

the final text line boundary, guided by the PDE. The numerical solution of (2) is based on a grid. First, an initial estimate of the function is calculated and, then, the function is updated on a discrete grid. Each update is also called one iteration.

### 4.2 Text Line Boundary Evolution

The level set method is a general framework. In order to achieve good results for a specified application, domain knowledge should be exploited. In this section, we discuss our modifications under the framework of the level set method for text line boundary evolution.

To evolve the text line boundary, we need an initial estimate of the text lines. Pixels with high density values allow a suitable initial estimate (see Fig. 2c). Global or local adaptive thresholding (such as the Niblack method [24]) can be used for this purpose. The initial estimate need not be very accurate. As long as at least one part of a text line is extracted, we can achieve a good final segmentation.

Text lines tend to be directional (for example, horizontal in our case). We can tune the level set method to exploit this a priori knowledge. In particular:

1. We force the boundary to grow faster in the horizontal direction by setting the horizontal update step larger than the vertical step. It is equivalent to compressing the image in the horizontal direction and updating the implicit function  $f$  by using the same step in both directions, but our approach is more efficient.
2. Curvature is adopted using a strategy called “the rich get richer”: The  $b_k$  in (2) is set to the square of actual curvature. By our assumption, text lines are horizontally elongated; therefore, curvatures of the left and right ends of boundaries are larger than those of the top and bottom, which means that the text line tends to grow faster in the horizontal direction.

In our experiments, the density estimation  $PDF_{est}$  serves as the normal speed  $S_N$ . Using  $PDF_{est}$  as the speed, the boundary will grow faster inside the text lines, where black pixel densities are large, and slower when it approaches gaps. Thus,  $PDF_{est}$  actually controls the speed of the boundary.

Another constraint that we want to enforce under the framework of the level set method is to allow merging of boundaries only in the horizontal direction. However, a merge in a vertical direction can happen after each iteration if nothing prevents it. To solve this problem, we need to detect the merge when the boundary evolves.

At time  $t$ , we retrieve the zero level set and we obtain  $N$  bounding boxes of closed region  $S_n^t$ ,  $n = 1, \dots, N$ . Suppose that, after iteration  $t+1$ , the zero level sets are  $S_n^{t+1}$ ,  $n = 1, \dots, M$ ,  $M \leq N$ . For each  $S_n^{t+1}$ , it must contain one or more regions of  $S_n^t$ 's since all regions are growing. Assume that  $S_i^{t+1}$  contains but is not limited to  $S_j^t$  and  $S_k^t$ . We adopt a geometry-based detection to evaluate  $S_j^t$  and  $S_k^t$ . A collision happens if  $S_j^t$  overlaps  $S_k^t$  in the horizontal direction. One reasonable criteria is

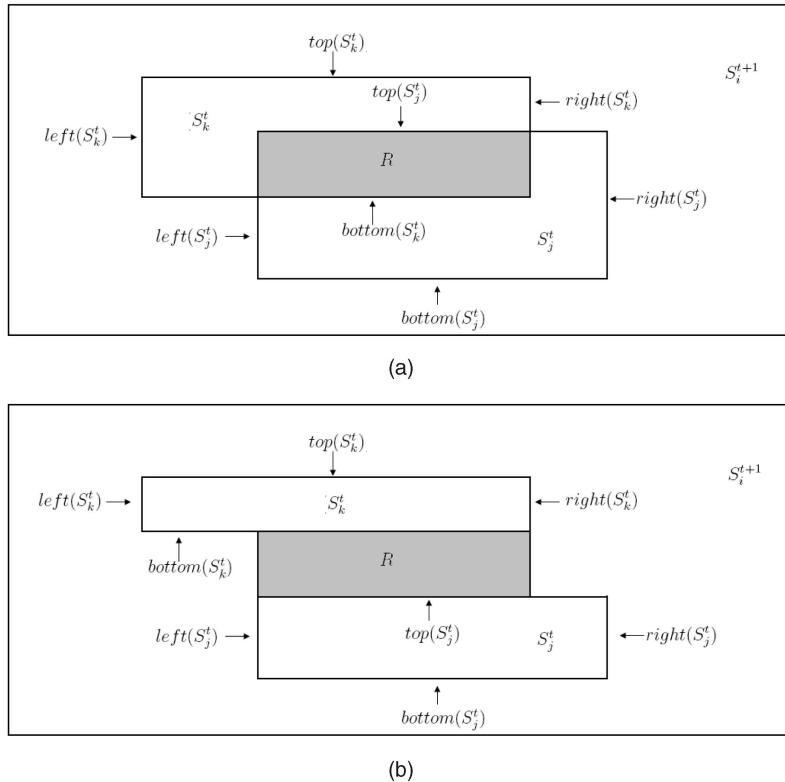


Fig. 3. The illustration of the collision.

$$left(S_k^t) \leq mean(S_j^t) \leq right(S_k^t) \quad (3)$$

or

$$left(S_j^t) \leq mean(S_k^t) \leq right(S_j^t), \quad (4)$$

where  $mean(S_k^t)$  is the  $x$  component of the center of gravity of  $S_k^t$ .  $right(S_k^t)$  and  $left(S_k^t)$  are the  $x$  components of the right and left ends of  $S_k^t$ , respectively.

If (3) or (4) holds, a collision happens. We then calculate the collision region  $R$  as follows: Without loss of generality, we assume that  $S_j^t$  is under  $S_k^t$  in the image, but two regions do not necessarily intersect (see Figs. 3a and 3b). That is,  $top(S_k^t) \leq top(S_j^t)$ , given that the origin is the upper left corner of the image. We define the boundary of region  $R$  as follows:

$$top(R) = min(bottom(S_k^t), top(S_j^t)), \quad (5)$$

$$bottom(R) = max(bottom(S_k^t), top(S_j^t)), \quad (6)$$

$$left(R) = max(left(S_k^t), left(S_j^t)), \quad (7)$$

$$right(R) = min(right(S_k^t), right(S_j^t)). \quad (8)$$

If collisions occur, we roll back to time  $t$  and we set  $S_N$  and  $b_k$  of the collision region  $R$  to zero. Since there is no force to evolve the boundaries, the two adjacent lines will not merge in this region in future iterations. Finally, we redo the evolution under the new configuration for iteration  $t + 1$ .

In summary, the procedure operates as follows:

1. Binarize  $PDF_{est}$  to obtain initial estimates  $LS_{initial}$  by using the Niblack algorithm [24].
2. Initialize the level set function  $f$  by using the initial estimate  $LS_{initial}$ . A simple initialization procedure calculates the distance transform of  $LS_{initial}$  as follows:

$$f = DIST(LS_{initial}^-) - DIST(LS_{initial}^+), \quad (9)$$

where  $LS_{initial}^+(x, y)$  and  $LS_{initial}^-(x, y)$  represent the regions inside and outside the boundary of  $LS_{initial}$  and  $DIST()$  is the distance transform [25] of the respective regions to the boundary. For a pixel inside the boundary,  $f < 0$ , and vice versa for a pixel outside the boundary. This is a standard process in the level set method [20].

3. Evolve the initial level set according to the PDE (2).
4. Perform vertical merge detection, as described above.
5. Repeat Steps 3 and 4 until the change between two iterations is small (result converges) or the number of iterations is larger than a predefined limit.

In our experiment, we found that 10 iterations are usually enough to achieve good results. For details of the level set method, refer to [23].

### 4.3 Postprocessing

After growing the initial estimate of text lines by using the level set method, most text line boundaries are correctly detected, but a few fragments may still exist (see Fig. 2d) due to extraordinarily large horizontal word gaps. These

**TABLE 1**  
Statistics on Components at Different Stages in Our Approach on a Set of Handwritten Chinese Documents

	Original image	After boundary evolving	After post-processing
Average length of components (in pixels)	16	336	398
Number of components	1,179	97	26

**TABLE 2**  
Some Statistics of the UMD Data Set

Script	Chinese	Cyrillic	Greek	Hebrew	Hindi	Japanese	Korean	Persian	Thai
Pages	1,148	546	15	117	2,721	1,121	1,483	68	309
Writer	74	5	1	4	91	8	245	1	10

fragmented pieces can easily be linked to obtain the final text lines by using a rule-based approach similar to [1].

We briefly describe our rule as follows:

1. Assign segments to unique IDs.
2. Calculate the orientation of each line segment by using the minimal mean squared error method.
3. Calculate the length of each line segment. If a text line is shorter than  $\frac{1}{10}$  of the page length, its orientation is assumed to be horizontal. If the length and the width of a connected component are both smaller than  $\frac{1}{20}$  of the page length, it is labeled as “isolated.” All text line segments that are not labeled as “isolated” are counted as “unprocessed.”
4. Sort the “unprocessed” text line segments by length.
5. Select the longest “unprocessed” text line segment and extend the line to left or right by using the following merging rules: we merge it to another “unprocessed” line segments if a) the difference in their orientations is less than 10 degrees, b) their bounding boxes’ horizontal gap is less than  $\frac{1}{10}$  of the page length, and c) their bounding boxes overlap more than 50 percent in the vertical direction. The ID of the merged segment is then updated. We iteratively apply this rule until there are no more “unprocessed” line segments in left or right, satisfying the merging conditions.
6. Mark all segments involved in Step 4 as “processed.”
7. Go to Step 5 until all line segments have been processed.

Finally, we group the isolated connected components to their closest major text line. We first estimate text line height from the histogram of the heights of the line segments’ bounding boxes. If the distance between the isolated connected components and the closest text line is less than the estimated text line height, we label these isolated components the same as the closest text line. Otherwise, we label those far from any major text line as noise. The estimated text line height need not be very accurate because the isolated components that belong to the characters are usually very close to the text lines (for example, diacritical marks). Fig. 2e shows the result after broken line segment linking and grouping the isolated connected components. Fig. 2f shows the final text line segmentation result by using different colors to distinguish neighboring text lines.

After text line boundary evolution using the level set method, linking of the remaining broken lines is easy and reliable because only a few fragmented line segments occur and the average length of a line segment is larger. Table 1 provides statistics about the number of components and average component length at each stage on a handwritten Chinese document set. After boundary evolution, the length of a line segment is almost 20 times that of the connected components on average. The number of components also significantly drops to approximately 1/10 the original number. We further reduce the number of components by grouping broken line segments and isolated small pieces.

## 5 EXPERIMENTS

### 5.1 Data Set

In this section, we present a quantitative analysis of our algorithm on a large database. The UMD test data set was collected by the members of the Language and Media Processing Laboratory at the University of Maryland.<sup>1</sup> Hard copies of handwritten documents were collected from many countries such as China, India, Korea, and Japan and were scanned at a resolution of 300 dots per inch (DPI) to binary TIFF images. In summary, 7,528 heterogeneous handwritten documents from 439 writers in nine scripts (Cyrillic, Greek, Hebrew, Hindi, Korean, Japanese, Persian, Chinese, and Thai) have been collected. Table 2 shows some statistics for each script. A typical image size is approximately 2,000 × 1,500 pixels and the character sizes vary from roughly 10 × 10 to 30 × 30 pixels. We also collected a large data set of Arabic documents containing 166,071 images. We have selected some for our experiments. Fig. 4 shows some examples.

### 5.2 Overview of the Experiments

We organize our experiments and evaluation as follows: First, we present our evaluation methodology and overall quantitative result on our database and, then, we quantitatively compare our method with other classical methods. To show the robustness of our algorithm, we rotate, scale, and add random noise to images to generate synthesized images. Segmented results are visualized using different colors which denote different text lines. Fig. 5 shows the

<sup>1</sup> The data set and the elevation algorithms are available for direct comparison by contacting the authors.

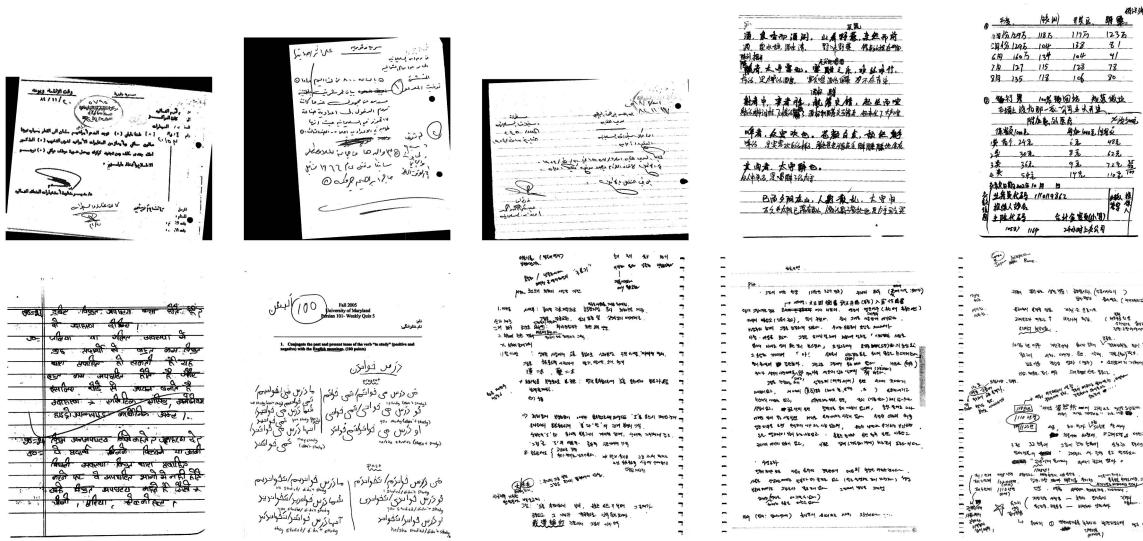


Fig. 4. Handwriting examples in the UMD data set.

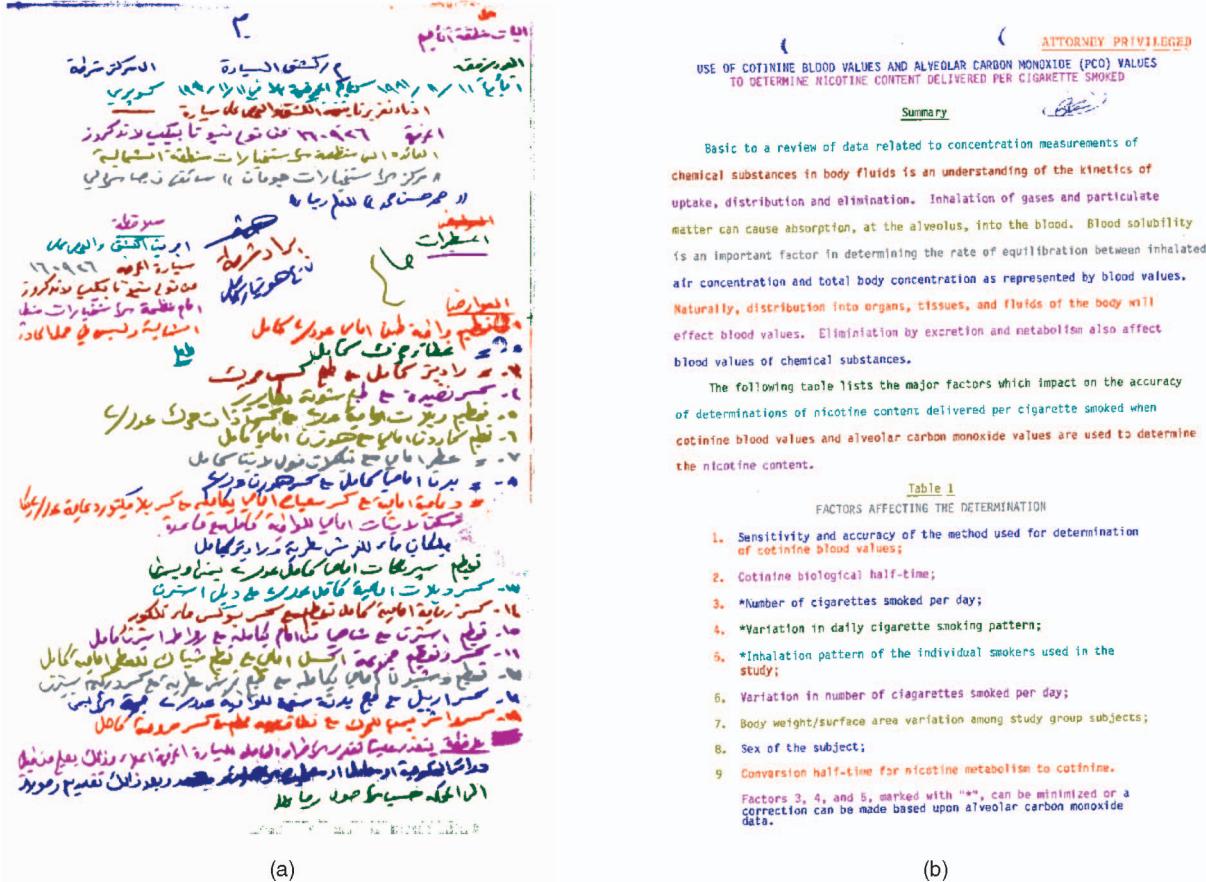


Fig. 5. (a) and (b) Our algorithm's results for Figs. 1a and 1c, respectively.

segmentation results of our algorithm on Figs. 1a and 1c. More examples appear in Fig. 6.

In all of these experiments, we use the same parameters. Specifically, we use the same Gaussian window and standard deviation for density estimation and the number of iterations for region growth is the same (10 iterations) as well. In our experiment, we set the standard deviations of a 2D Gaussian kernel to 10 and 40 pixels in the horizontal and vertical directions, respectively. Therefore, the size of the

truncated window is  $30 \times 120$  pixels according to the discussion in Section 3.2. We also demonstrate how these parameters influence the result at the end of this section and give the best window size for the current data set.

Four scripts—Arabic, Chinese, Hindi, and Korean—are used to illustrate the performance of different methods in Section 5.3. We have ground truthed 100 documents per script by using the VIPER Ground Truth Editor [26]. Since bounding boxes are not sufficiently accurate to delineate the

年龄	性别	姓名	年龄	性别	姓名
18-25	男	王立群	17-25	女	王立群
26-35	男	王立群	16-25	女	王立群
36-45	男	王立群	17-25	女	王立群
46-55	男	王立群	17-25	女	王立群
56-65	男	王立群	17-25	女	王立群
66-75	男	王立群	17-25	女	王立群
76-85	男	王立群	17-25	女	王立群
86-95	男	王立群	17-25	女	王立群
96-105	男	王立群	17-25	女	王立群

Table 1 Factors Affecting the Determination

1. Sensitivity and accuracy of the method used for determination of cotinine blood values;

Table 2 Factors Affecting the Determination

1. Cotinine biological half-time;

2. Number of cigarettes smoked per day;

3. Variation in daily cigarette smoking pattern;

4. Inhalation pattern of the individual smokers used in the study;

5. Variation in number of cigarettes smoked per day;

6. Body weight/surface area variation among study group subjects;

7. Sex of the subject;

8. Conversion half-time for nicotine metabolism to cotinine.

Factors 3, 4, and 5, marked with \*\*+, can be minimized or a correction can be made based upon alveolar carbon monoxide data.

1159202611

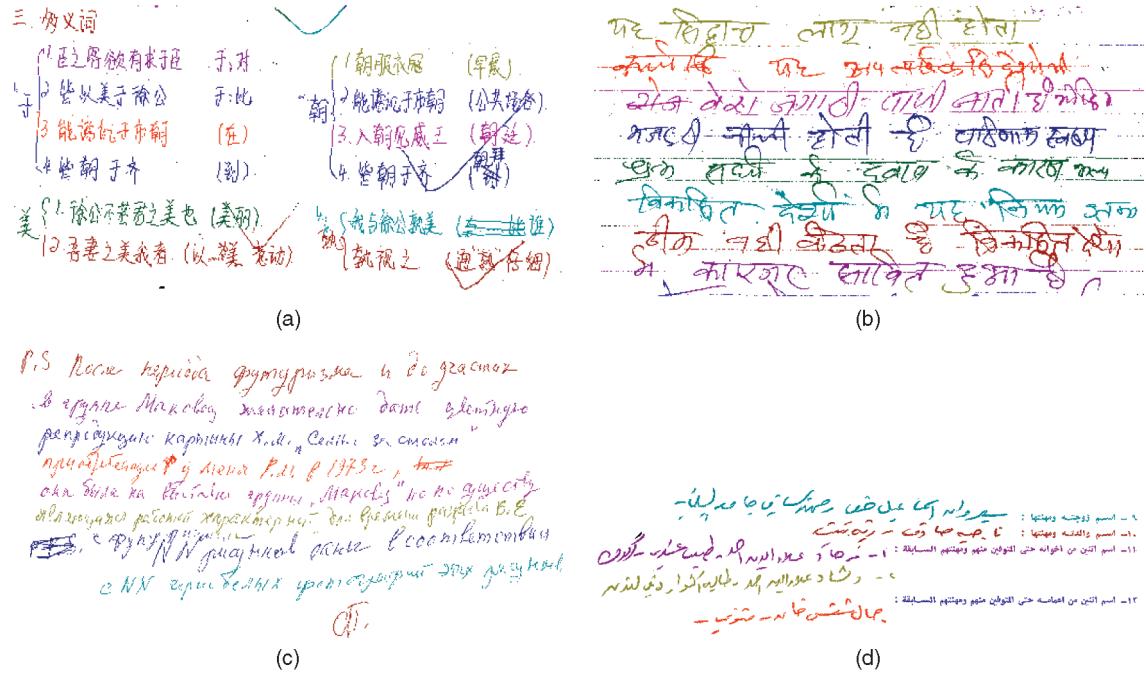


Fig. 6. Color examples of text line segmentation results. (a) Handwriting with an irregular document layout. (b) Handwritten text lines with rule lines. (c) Curvilinear handwritten text lines. (d) Document mixed with handwritten and machine-printed text lines.

boundary of text lines, we use polygons for the ground truth. In total, there are 7,660 handwritten text lines.

### 5.3 Evaluation Methodology

Traditionally in machine printed document analysis, the text line segmentation results are represented as rectangular bounding boxes. Therefore, the evaluation is often based on the four coordinates of a bounding box. However, some overlaps inevitably occur among bounding boxes in curvilinear handwritten documents. Nonoverlapping closed curves are better in the sense of representing curvilinear handwritten text lines. By this representation, the evaluation can be done at the pixel level, which is more accurate than that done at the bounding box level.

Supposing there are  $M$  ground-truthed lines and  $N$  detected lines, we construct an  $M \times N$  matrix  $P$ . An element  $P_{i,j}$  for  $i = 1, \dots, M$  and  $j = 1, \dots, N$  of matrix  $P$  represents the number of shared black pixels between the  $i$ th ground-truthed text line and the  $j$ th detected line. We enforce one-to-one correspondence between the detected lines and the ground truth. Since the number of lines in two sets are different in general, we augment the matrix  $P$  to a square matrix  $P'$ . A line is allowed to be matched to a dummy line and this match has no shared pixels ( $P_{i,j} = 0$ ). The square matrix  $P'$  has a dimension  $\max(M, N) \times \max(M, N)$ . For each assignment of the correspondence for ground truth  $S(k)$ ,  $k = 1, \dots, \max(M, N)$ , the goodness  $G(S)$  of this assignment is the total number of shared black pixels:

$$G(S) = \sum_{k=0}^{\max(M,N)} P_{k,S(k)}. \quad (10)$$

The best assignment  $S_o$  is the one with maximum goodness:

$$S_o = \arg \max_S G(S). \quad (11)$$

The Hungarian algorithm [27] is used to efficiently search for the assignment problem.

The overall pixel-level hit rate is defined as

$$H = \frac{G(S_o)}{\text{number of black pixels in the ground truth}}. \quad (12)$$

By using the pixel-level hit rate and the Hungarian algorithm, different segmentation errors, for example, splitting, merging, and missing, can be appropriately penalized with weights that are proportional to the number of pixels involved.

We can also evaluate the performance at the text line level. One ground-truthed line  $i$  is claimed to be correctly detected if

$$\frac{P_{i,S_o(i)}}{\sum_{j=1}^N P_{i,j}} \geq 0.9 \quad (13)$$

and

$$\frac{P_{i,S_o(i)}}{\sum_{k=1}^M P_{k,S_o(i)}} \geq 0.9. \quad (14)$$

In other words, if a ground-truthed line and the corresponding detected line share at least 90 percent of the pixels with respect to both of them, a text line is claimed to be correctly detected. In the following experiments, we evaluate the performance based on both the pixel-level hit rate and the text-line-level detection rate.

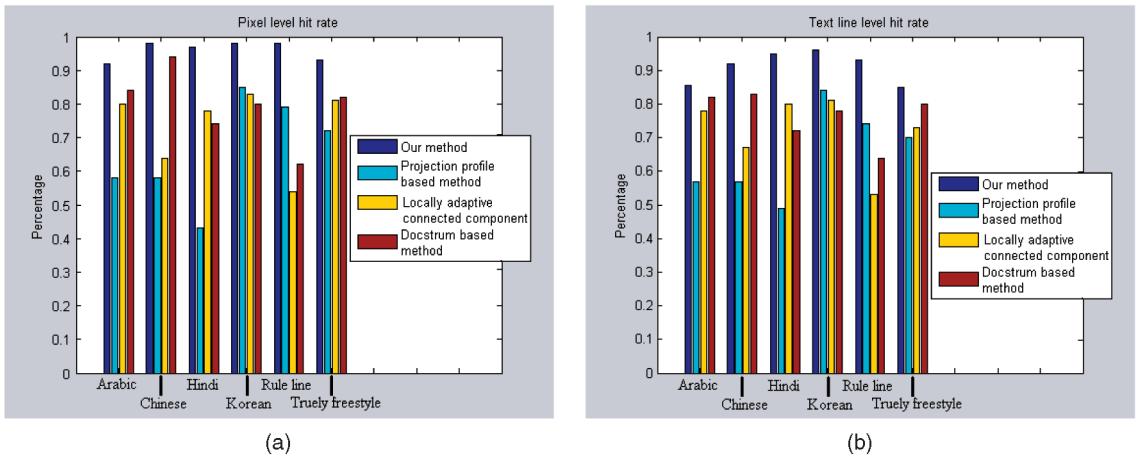


Fig. 7. Bar charts of (a) pixel-level hit rates and (b) text-line-level detection rates of different algorithms on different data sets.

TABLE 3  
Quantitative Comparison of 100 Handwritten Arabic Documents with a Total of 2,691 Text Lines

	Pixel-level hit rate	STD of pixel-level hit rate	Detected text lines
Our method	92%	0.04	2,303 (85.6%)
Projection profile based method [3]	58%	0.17	1,319 (57%)
Locally adaptive connected component [2]	80%	0.11	1,799 (78%)
Docstrum based method [1]	84%	0.09	1,904 (82%)

TABLE 4  
Quantitative Comparison of 100 Handwritten Chinese Documents with a Total of 1,672 Text Lines

	Pixel-level hit rate	STD of pixel-level hit rate	Detected text lines
Our Method	98%	0.01	1,532 (92%)
Projection profile based method [3]	58%	0.24	965 (57%)
Locally adaptive connected component [2]	64%	0.12	1,131 (67%)
Docstrum based method [1]	94%	0.04	1,389 (83%)

## 5.4 Performance and Comparison

We organize our discussion on performance and comparison as follows: First, we briefly show the accuracy on four scripts and, then, we discuss the performance on two different categories, that is, handwriting with and without rule lines. After that, we compare our performance with three other techniques. As a top-down method, the X-Y cut [3] builds a structural tree that represents the document layout. The projection histogram analysis determines the cut in each step. Docstrum [1], on the other hand, builds the layout structure in a bottom-up manner from the connected components. We can therefore retrieve the text line level from the tree structure, although these methods are not solely designed for text line segmentation. Our comparison experiments use a public domain implementation [28] of the X-Y cut and Docstrum algorithms. In addition to these two classical methods, we also test a locally adaptive connected component-based method in DOCLIB [2]. Since the text line segmentation in handwritten documents has not been sufficiently addressed in previous work, we can only compare our approach to those methods that were originally designed for machine printed documents.

In principle, we can train the optimal parameters of an algorithm based on ground truth, as Mao and Kanungo did in [28] for the evaluation of different page segmentation algorithms. As shown in their paper, a robust algorithm often works well under a wide range of parameter inputs. Therefore, in our experiments, we manually tuned parameters for all algorithms based on a few training images and then fixed these parameters for all of the experiments.

### 5.4.1 Performance of Our Algorithm

Our method's performance for different scripts is shown in Fig. 7, with details listed on the first rows of Tables 3, 4, 5, and 6. Overall, we achieve a 94.7 percent accuracy in terms of the pixel-level hit rate. The standard deviation of the accuracy on all scripts is small, which shows that our algorithm is stable. At the text line level, there are a total of 7,660 ground-truthed lines in Tables 3, 4, 5, and 6 and our approach can correctly detect 6,982 (91.2 percent) of them. Details are shown under the last columns in Tables 3, 4, 5, and 6. Our algorithm does not use script-dependent information and we set the same parameter for all scripts. A slight degradation in performance is observed on the Arabic documents because they are less constrained (for

TABLE 5  
Quantitative Comparison of 100 Handwritten Hindi Documents with a Total of 1,365 Text Lines

	Pixel-level hit rate	STD of pixel-level hit rate	Detected text lines
Our method	97%	0.02	1,295 (95%)
Projection profile based method [3]	43%	0.37	673 (49%)
Locally adaptive connected component [2]	78%	0.19	1,103 (80%)
Docstrum based method [1]	74%	0.22	996 (72%)

TABLE 6  
Quantitative Comparison of 100 Handwritten Korean Documents with a Total of 1,932 Text Lines

	Pixel-level hit rate	STD of pixel-level hit rate	Detected text lines
Our method	98%	0.01	1,852 (96%)
Projection profile based method [3]	85%	0.13	1,633 (84%)
Locally adaptive connected component [2]	83%	0.13	1,572 (81%)
Docstrum based method [1]	80%	0.14	1,499 (78%)

example, there might not exist consistent base lines for handwritten Arabic text lines (see Fig. 1a)). Due to the inherent differences among scripts, the performances of different scripts could also be slightly different.

Not all handwritten documents are truly freestyle and written on blank sheets (for example, Fig. 8a). Many, such

as lecture notes and some drafts, are written on papers with rule lines (for example, Fig. 8c) and they are more or less constrained by the rule lines. Therefore, we evaluate our performance on both categories. Generally, rule lines make the handwritten text lines straight, but they may also degrade the performance of the connected component analysis because the rule lines may connect arbitrary numbers of adjacent characters. The projection profile analysis may benefit from the rule lines. However, we do not know the geometric relationship between handwritten text lines and rule lines. It may cause errors in profile analysis. For example, writers may ignore the rule lines. As another example, most Chinese handwritings are written over the rule lines, while Hindi tend to overlap the top stroke with the rule lines in our collected database. To evaluate the performance of our algorithm on these two categories, we prepared two sets of documents with and without rule lines, each containing 100 documents chosen from all four scripts. The accuracy of our algorithm on these two mixed sets is shown in the second rows of Tables 7 and 8, respectively. At the text line level, our approach can detect 1,466 out of 1,589 (93 percent) for documents with rule lines and 1,863 out of 2,178 (85 percent) for freestyle documents.

Generally, the majority of failures happen because two neighboring text lines significantly touch each other (for example, see Fig. 9). Since two lines are connected in only a few touched areas, postprocessing to segment them is necessary to improve the performance. We also notice that the correction in the gap between two lines (Fig. 9a) may create many peaks in the density function estimation and it is not trivial to segment them. Other challenges include nontext objects such as stamps, logos, and severe noise. We will measure the effect of noise at the end of this section.

#### 5.4.2 Comparison with Other Methods

In this section, we compare our method with three other methods. Some previous work presented the accuracies of their methods at the text line level, but none measured the accuracy at the pixel level. The results of most traditional

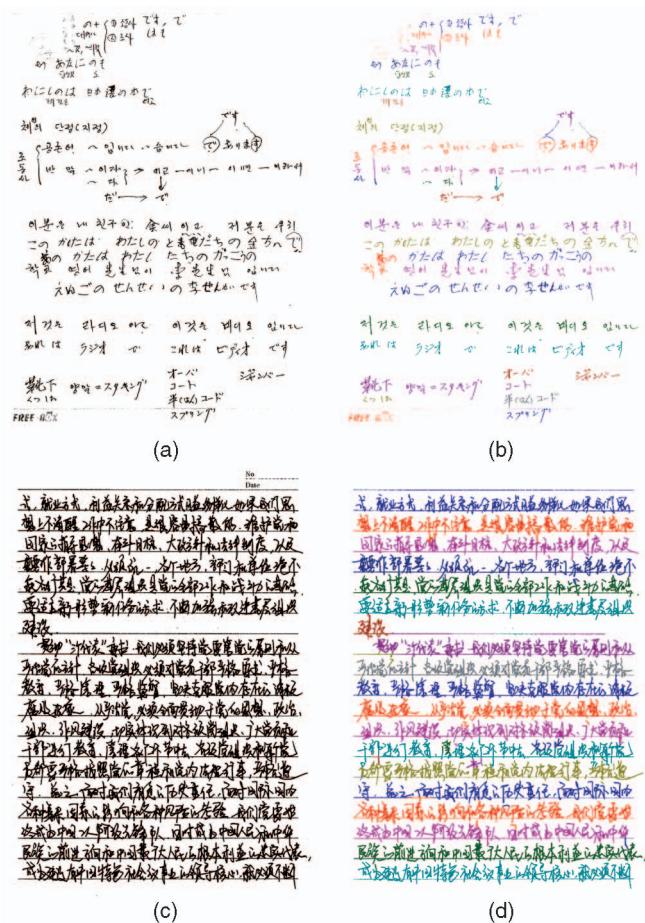


Fig. 8. (a) A freestyle document image. (b) Results of our algorithm for (a). (c) Document image with rule lines. (d) Results of our algorithm for (c).

TABLE 7  
Quantitative Comparison of 100 Handwritten Documents with Rule Lines with a Total of 1,589 Text Lines

	Pixel-level hit rate	STD of pixel-level hit rate	Detected text lines
Our method	98%	0.01	1,466 (93%)
Projection profile based method [3]	79%	0.10	1,178 (74%)
Locally adaptive connected component [2]	54%	0.34	849 (53%)
Docstrum based method [1]	62%	0.29	1,024 (64%)

TABLE 8  
Quantitative Comparison of 100 Freestyle Handwritten Documents with a Total of 2,178 Text Lines

	Pixel-level hit rate	STD of pixel-level hit rate	Detected text lines
Our method	93%	0.03	1,863 (85%)
Projection profile based method [3]	72%	0.16	1,537 (70%)
Locally adaptive connected component [2]	81%	0.13	1,576 (73%)
Docstrum based method [1]	82%	0.15	1,751 (80%)

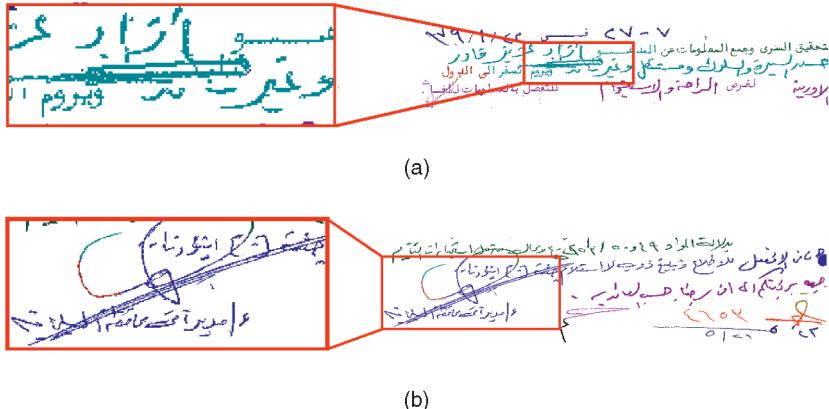


Fig. 9. Failure examples of our method and their close-ups of the segmentation results.

methods use bounding boxes, which may overlap each other and cause confusion. To make pixel-level evaluation possible, we group pixels in the overlapping area of bounding boxes to the closest text line.

Statistical results for different methods are shown in Fig. 7, with details listed in the second through fourth rows of Tables 3, 4, 5, and 6. We achieve much better results in comparison to other methods, regardless of the scripts used. For example, the pixel-level hit rate of the connected component-based method is 78 percent in the Hindi data set (Table 5), while our method is 97 percent. At the text line level, these methods perform significantly worse. Table 5 shows that only 996 (43 percent) text lines are correctly detected using the projection profile method in Hindi documents, which is significantly worse than our method (95 percent). This is primarily because the top line of the Hindi words and the rule lines introduce many false peaks in the projection profile analysis. Generally, our algorithm is quite stable for all four scripts.

Table 8 shows that the two connected component-based methods perform better on freestyle handwritten documents such as the Arabic and the Hindi, while the projection profile-based method is better on documents with rule lines (Table 7). Comparing the same algorithm in

two categories, we observe that the rule lines slightly improve the performance of the projection profile-based method, but they seriously deteriorate the performance of connected component-based methods. We found that connected component-based methods rely on some geometric measures, such as the size of connected components, to filter spurious components (such as small noise dots and big figures on the document) and guide the grouping of neighboring components. The nice property of these geometric measures is destroyed by rule lines. On the contrary, our algorithm does not filter anything before we detect the regions. Projection profile-based methods do not need to find connected components with suitable size as well, but they suffer with complex page layouts and curvilinear lines. Examples are shown in Figs. 10c and 10d. Text lines are often broken or incorrectly merged. In our result (for example, Fig. 10a), most text lines are clearly labeled without confusion.

A good text line segmentation algorithm should process machine-printed documents as well. Fig. 5b shows the segmentation results on a machine printed document without tuning parameters. It shows that our algorithm performs better than or comparably to traditional methods on machine printed documents.

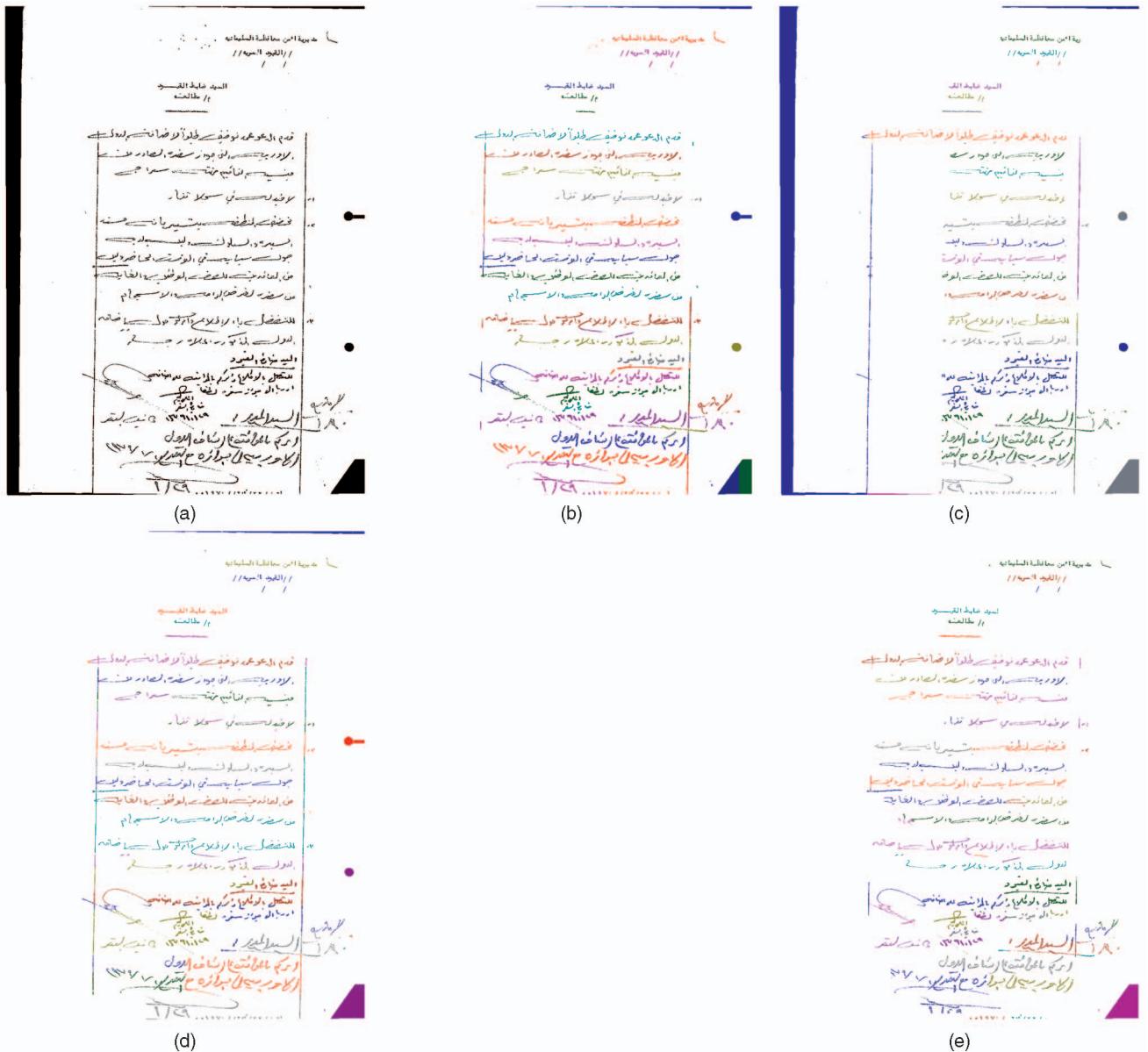


Fig. 10. Comparison with classical methods. (a) The original image. (b) Our method. (c) Docstrum method [1]. (d) X-Y cut method [3]. (e) Locally adaptive connected component-based method [2].

The connected component-based and projection profile-based methods are useful for determining the orientation of zones, size of characters, etc. However they do not work well for handwritten text line segmentation. This comprehensive experiment demonstrates that our method can be used for handwritten documents, machine printed documents, and documents that are a mix of machine printed and handwritten text. Our assumption fits well for both machine printed and handwritten text lines. The statistical results show that our method is better in handwritten documents than classical methods and their variations.

## 5.5 Robustness Test

In this section, we examine the robustness of our algorithm to both changes in internal parameters and changes in the input. First, the internal parameters are changed in a wide range. Then, the external input is varied by rotation and

scale. We then present the results on synthesized images with different noise levels.

### 5.5.1 Robustness with Respect to Internal Parameters

Two major components comprise our algorithm: the Parzen window-based density estimation and boundary evolution. The region's evolutions become slow in the text line boundaries due to the low value of the PDF and, if a region has a zero value, the boundary will stop at the region. Our approach is very stable under a reasonable number of iterations. In experiments, we did not find a significant difference, even if we increase the iteration number from 10 to 50. We therefore evaluate the Parzen window size in the following.

We change the window size while keeping the standard deviations proportional to the window size. This way, we evaluate the performance of internal parameters. Fig. 11 shows the performance of different Parzen windows. The

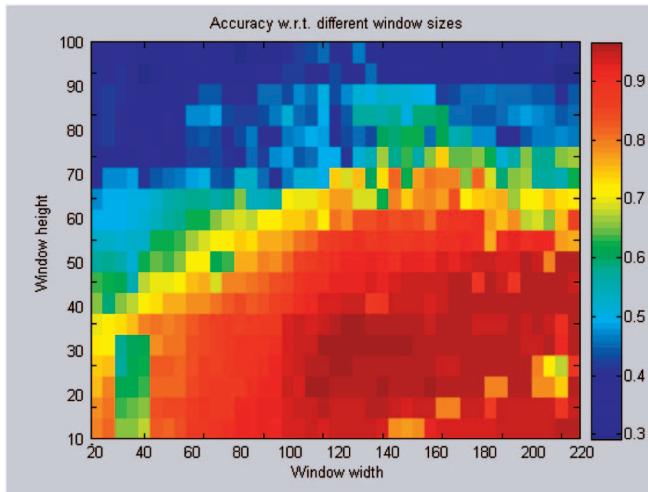


Fig. 11. Pixel-level hit rate of our method under the change in window size.

accuracy is visualized using a color map. The  $x$  axis is the width of the kernel window, while the  $y$  axis is the height of the kernel window. The best size of the kernel window is  $30 \times 120$  in our case. In all of our experiments, we maintain these values.

### 5.5.2 Performance under Scale and Rotation Changes

Our algorithm is not theoretically scale invariant or rotation invariant. In the following experiments, we test our method on synthesized images under the changes in scale and rotation on the four annotated data sets. The character size within images is affected by 1) the scanning resolution and

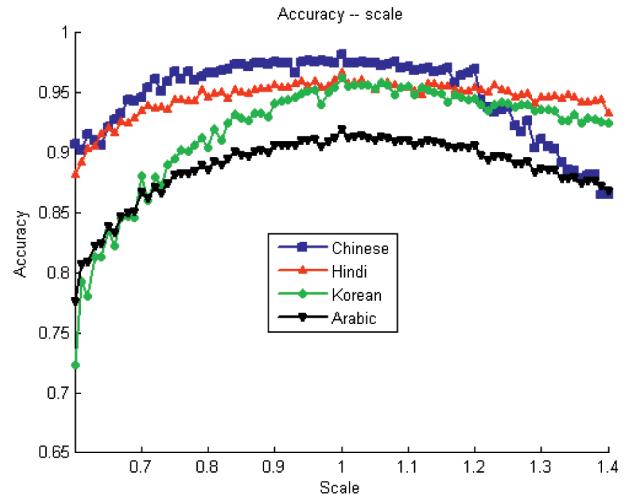


Fig. 12. Pixel-level hit rate of our method under the change in scale.

2) the font size. Therefore, we can evaluate the sensitivity of our method by resizing the image to mimic the change in character size. Fig. 12 shows the accuracy for different scales. It shows that our algorithm produces similar results with different character sizes. One example appears in Fig. 13.

To illustrate how the skew degrades the performance, we rotate an image using different angles and compare the accuracy. Fig. 14 shows the quantitative evaluation under different rotation angles and Fig. 15 shows one example under three different rotation angles. The performance of our algorithm does not significantly degrade when the orientation changes within  $[-10, 10]$  degrees. For most deskewed handwritten documents, the skew falls into this

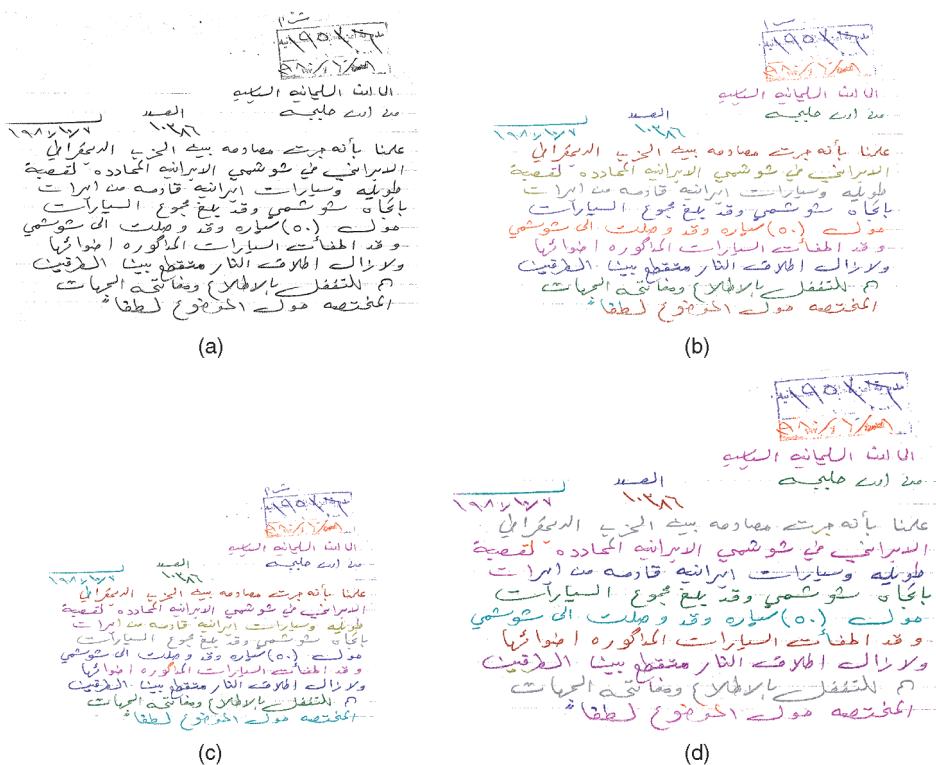


Fig. 13. Segmentation results for documents with different scales. (a) Original image. (b) Result on the original image. (c) Result on the image scaled to 80 percent. (d) Result on the image scaled to 120 percent.

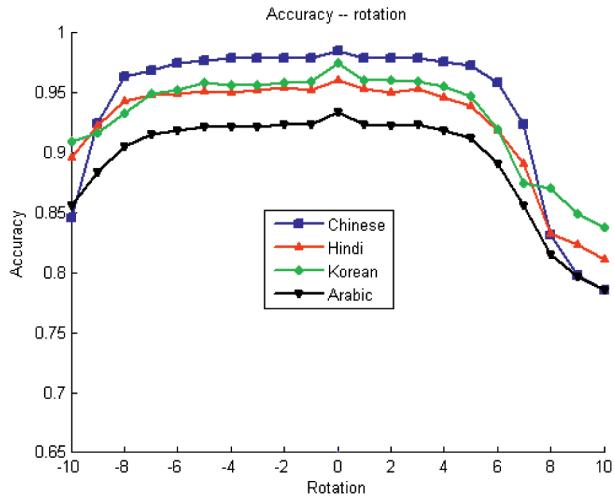


Fig. 14. Performance of our method under the change in rotation angles.

range. Experiments show that our algorithm can fit, in most cases, with a deskew module as preprocessing.

By using the best internal parameters, as discussed in Section 5.5.1, we observe small peaks in Figs. 12 and 14. Overall, our performance shows that it can tolerate input changes to some extent. The density estimation is not sensitive to scale and rotation. Our algorithm is designed for curvilinear text lines, so it can still output the expected result when the rotation is not significant (for example,  $\pm 10^\circ$ ).

### 5.5.3 Performance under Noise

Noise is a major issue in many computer vision problems. It can significantly degrade the result. In our method, it mainly leads to inaccurate density estimation. In principle, a good noise model can be integrated in the density estimation module to compensate for the degradation and improve the accuracy.

In this experiment, we demonstrate the performance by using salt-and-pepper noise. By randomly flipping the pixels in binary images, we can perform simulations with different

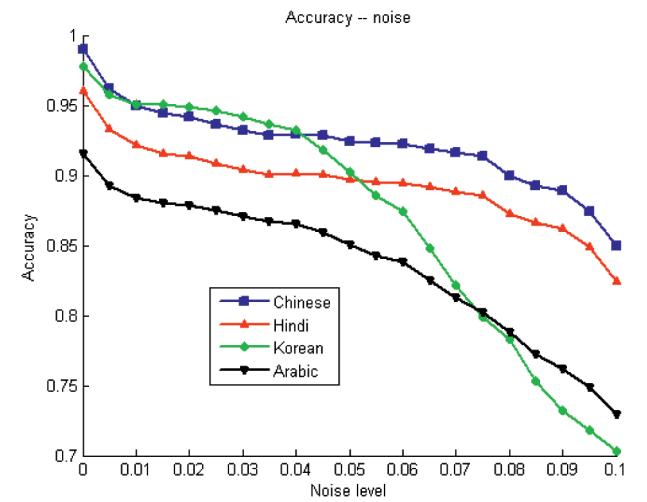


Fig. 16. Performance of our method under the change in noise levels.

probabilities. Fig. 17 shows three images with different noise levels. Fig. 16 gives a statistical analysis of accuracy in terms of different noise probabilities for the four scripts.

Results show that our algorithm elegantly degrades under noise and noise reduction and text restoration [29], [30], [31] are helpful but are beyond the scope of our paper.

We have implemented our algorithm in C++. The average processing time for an image with the size of approximately  $2,000 \times 1,500$  pixels nears 20 s on a PC with a 1.6 GHz CPU and 1-Gbyte memory. Our algorithm is relatively slow compared to other algorithms. The most time-consuming part of our algorithm involves the computation of the partial derivatives in boundary evolution. A more efficient implementation with the fast marching methods [32] may significantly speed up the algorithm.

## 6 DISCUSSION, CONCLUSION, AND FUTURE WORK

In this paper, we proposed a novel and robust text line segmentation approach for freestyle handwritten documents. Extensive comparison experiments show that the

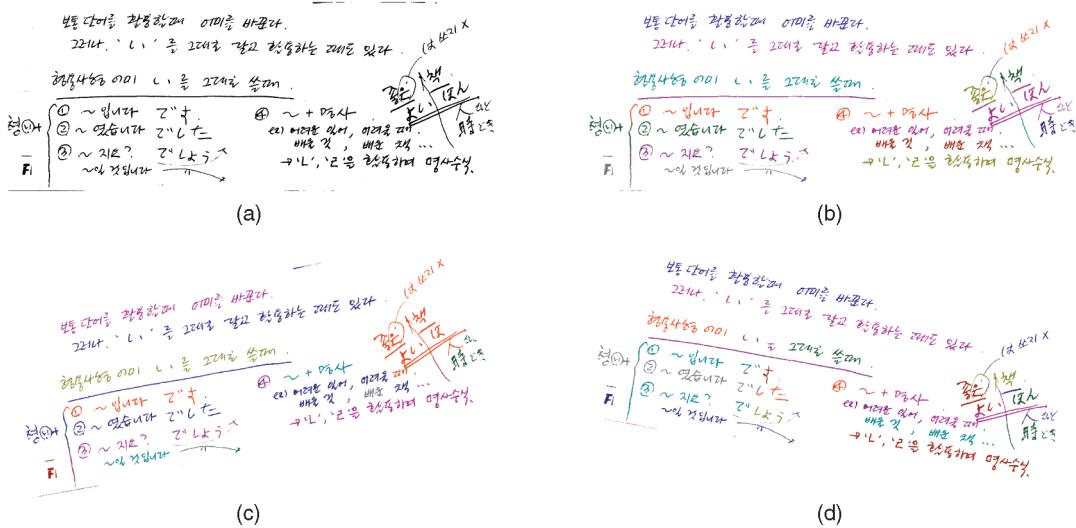


Fig. 15. Segmentation results for documents with different rotations. (a) Original image. (b) Result on the original image. (c) Result on the image rotated  $-10^\circ$ . (d) Result on the image rotated  $10^\circ$ .

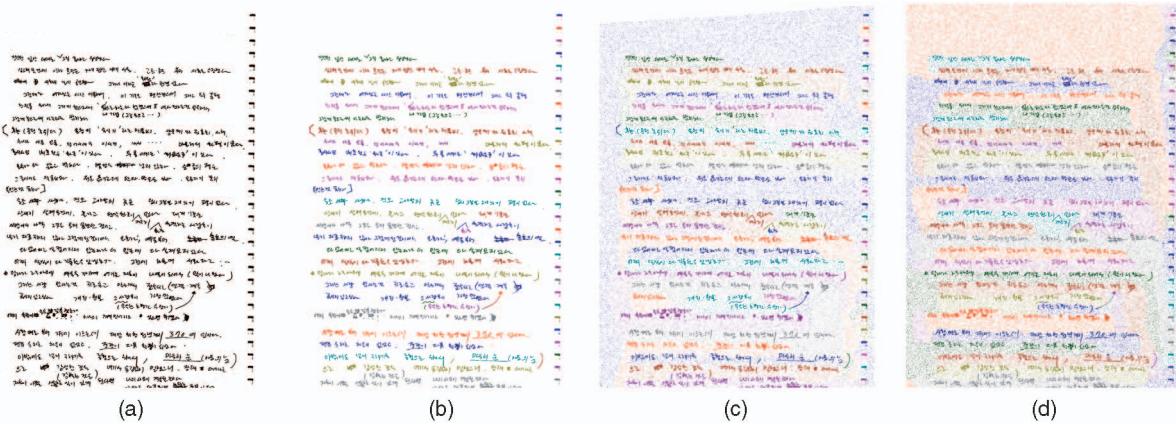


Fig. 17. Segmentation results for documents with different image quality. (a) Original image. (b) Result on the original image. (c) Result on the image with 5 percent noise. (d) Result on the image with 10 percent noise.

proposed method consistently outperforms the traditional text line segmentation methods such as the projection-based top-down approaches and connected component-based bottom-up approaches. Our approach is script independent since no script-specific knowledge is used in our approach. We achieve consistently high accuracy on many scripts, such as Arabic, Hindi, Chinese, and Korean, by using the same parameter settings. Experiments show that our algorithm is robust and it elegantly degrades under the changes in scales, rotation, and noise. Although it was originally developed with a binary image in mind, our algorithm can extend to gray-scale and color document images without major changes. The time complexity for processing gray-scale images is the same as that for binary ones. For color images, the time complexity is three times that for binary ones in the probability map estimation because the algorithm needs to combine three channels, while it is the same in the remaining processing steps.

Our approach combines the advantages of both the bottom-up approaches and the top-down ones. First, our approach does not require the layout information a priori, having the same advantage as many bottom-up methods. Based on the probability map of the text lines, the boundaries automatically grow, merge, and stop. Therefore, we can deal with complicated document layouts. Second, we do not assume particular scripts as input, which is an attractive advantage, as in many top-down methods. We consider pixels, instead of characters, in our approach and estimate the unknown PDF of every document image. Therefore, our approach is script independent.

We observed that adjacent text lines may merge in our method if they significantly touch each other. This wrong merge can be detected because the segmented text line has an abnormal height. Further postprocessing can be developed to detect the touching area and separate the merged text lines. Under severe noise, the performance of the proposed method deteriorates. Document enhancement techniques may be exploited to improve the performance.

## ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for their constructive comments. This research was supported by the US Department of Defense under Contract MDA-9040-2C-0406.

## REFERENCES

- [1] L. O'Gorman, "The Document Spectrum for Page Layout Analysis," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 15, no. 11, pp. 1162-1173, Nov. 1993.
- [2] S. Jaeger, G. Zhu, D. Doermann, K. Chen, and S. Sampat, "DOCLIB: A Software Library for Document Processing," *Proc. SPIE Document Recognition and Retrieval XIII*, pp. 63-71, 2006.
- [3] G. Nagy, S. Seth, and M. Viswanathan, "A Prototype Document Image Analysis System for Technical Journals," *Computer*, vol. 25, no. 7, pp. 10-22, July 1992.
- [4] R. Plamondon and S.N. Srihari, "Online and Offline Handwriting Recognition: A Comprehensive Survey," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 22, no. 1, pp. 63-84, Jan. 2000.
- [5] Y. Li, Y. Zheng, and D. Doermann, "Detecting Text Lines in Handwritten Documents," *Proc. 18th Int'l Conf. Pattern Recognition*, pp. 1030-1033, 2006.
- [6] Y. Li, Y. Zheng, D. Doermann, and S. Jaeger, "A New Algorithm for Detecting Text Line in Handwritten Documents," *Proc. 10th Int'l Workshop Frontiers in Handwriting Recognition*, pp. 35-40, 2006.
- [7] S. Mao, A. Rosenfeld, and T. Kanungo, "Document Structure Analysis Algorithms: A Literature Survey," *Proc. SPIE Document Recognition and Retrieval X*, pp. 197-207, 2003.
- [8] A.K. Jain and B. Yu, "Document Representation and Its Application to Page Decomposition," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 20, no. 3, pp. 294-308, Mar. 1998.
- [9] J. Liang, I.T. Phillips, and R.M. Haralick, "Performance Evaluation of Document Structure Extraction Algorithms," *Computer Vision and Image Understanding*, vol. 84, no. 1, pp. 144-159, 2001.
- [10] G. Nagy, "Twenty Years of Document Image Analysis in PAMI," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 22, no. 1, pp. 38-62, Jan. 2000.
- [11] A. Simon, J.-C. Pret, and A. Johnson, "A Fast Algorithm for Bottom-Up Document Layout Analysis," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 19, no. 3, pp. 273-277, Mar. 1997.
- [12] B. Yu and A. Jain, "A Robust and Fast Skew Detection Algorithm for Generic Documents," *Pattern Recognition*, vol. 29, no. 10, pp. 1599-1629, 1996.
- [13] R. Manmatha and J.L. Rothfeder, "A Scale Space Approach for Automatically Segmenting Words from Historical Handwritten Documents," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 27, no. 8, pp. 1212-1225, Aug. 2005.
- [14] U.-V. Martin and H. Bunke, "Text Line Segmentation and Word Recognition in a System for General Writer Independent Handwriting Recognition," *Proc. Sixth Int'l Conf. Document Analysis and Recognition*, pp. 159-163, 2001.
- [15] N. Tripathy and U. Pal, "Handwriting Segmentation of Unconstrained Oriya Text," *Proc. Ninth Int'l Workshop Frontiers in Handwriting Recognition*, pp. 306-311, 2004.
- [16] U. Pal and P.P. Roy, "Multioriented and Curved Text Lines Extraction from Indian Documents," *IEEE Trans. Systems, Man, and Cybernetics—Part B: Cybernetics*, vol. 34, no. 4, pp. 1676-1684, 2004.

- [17] A. Zahour, B. Taconet, P. Mercy, and S. Ramdane, "Arabic Handwritten Text-Line Extraction," *Proc. Sixth Int'l Conf. Document Analysis and Recognition*, pp. 281-285, 2001.
- [18] Y. Zheng, H. Li, and D. Doermann, "A Parallel-Line Detection Algorithm Based on HMM Decoding," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 27, no. 5, pp. 777-792, May 2005.
- [19] J. Sauvola, D. Doermann, and M. Pietikainen, "Locally Adaptive Document Skew Detection," *Proc. SPIE Document Recognition and Retrieval IV*, pp. 96-108, 1997.
- [20] S. Osher and R. Fedkiw, *Level Set Methods and Dynamic Implicit Surfaces*. Springer, 2003.
- [21] E. Parzen, "On Estimation of a Probability Density Function and Mode," *Annals Math. Statistics*, vol. 33, no. 3, pp. 1065-1076, 1962.
- [22] R.O. Duda, P.E. Hart, and D.G. Stork, *Pattern Classification*, second ed., pp. 84-97. Wiley-Interscience Publications, 2000.
- [23] B. Sumengen, "Variational Image Segmentation and Curve Evolution on Natural Images," PhD dissertation, Univ. of California, Santa Barbara, Sept. 2004.
- [24] W. Niblack, *An Introduction to Digital Image Processing*, pp. 115-116. Prentice Hall, 1986.
- [25] A. Rosenfeld and J. Pfaltz, "Sequential Operations in Digital Picture Processing," *J. ACM*, vol. 13, no. 4, pp. 471-494, 1966.
- [26] D. Doermann and D. Mihalcik, "Tools and Techniques for Video Performances Evaluation," *Proc. 15th Int'l Conf. Pattern Recognition*, pp. 167-170, 2000.
- [27] G. Liu and R.M. Haralick, "Optimal Matching Problem in Detection and Recognition Performance Evaluation," *Pattern Recognition*, vol. 35, no. 3, pp. 2125-2139, 2002.
- [28] S. Mao and T. Kanungo, "Software Architecture of PSET: A Page Segmentation Evaluation Toolkit," *Int'l J. Document Analysis and Recognition*, vol. 4, no. 3, pp. 205-217, 2002.
- [29] H.S. Baird, "Calibration of Document Image Defect Models," *Proc. Second Symp. Document Analysis and Information Retrieval*, pp. 1-16, 1993.
- [30] T. Kanungo, R.M. Haralick, and I. Phillips, "Nonlinear Local and Global Document Degradation Models," *Int'l J. Imaging Systems and Technology*, vol. 5, no. 4, pp. 220-230, 1994.
- [31] M. Cannon, J. Hochberg, and P. Kelly, "Quality Assessment and Restoration of Typewritten Document Images," *Int'l J. Document Analysis and Recognition*, vol. 2, nos. 2-3, pp. 80-89, 1999.
- [32] J. Sethian, *Level Set Methods and Fast Marching Methods, Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science*. Cambridge Univ. Press, 1999.



**Yi Li** received the BEng and MEng degrees from South China University of Technology, Guangzhou, China, in 2001 and 2004, respectively. He is currently working toward the PhD degree in the Department of Electrical and Computer Engineering at the University of Maryland, College Park. He is also with the Maryland Terrapin Team, which received Second Place in the First Semantic Robot Vision Challenge (SRVC) at the 22nd Conference on Artificial Intelligence (AAAI 2007) sponsored by the US National Science Foundation. His research interests include document image processing, shape analysis, object categorization, robotics, and computer vision. He is the Future Faculty Fellow of the A. James Clark School of Engineering at the University of Maryland. He received the Best Student Paper Award from the 10th International Workshop on Frontiers in Handwriting Recognition (IWFHR 2006). He is a student member of the IEEE.



**Yefeng Zheng** received the BE and ME degrees from Tsinghua University, Beijing, in 1998 and 2001, respectively, and the PhD degree from the University of Maryland, College Park, in 2005. He is currently a research scientist at Siemens Corporate Research, working on medical image analysis. His research interests include image analysis, pattern recognition, and computer vision. As a codeveloper of an Asian OCR system, he received the National Scientific and Technological Progress Award (Second Class) of China in 2003. He is a member of the IEEE.



**David Doermann** received the BSc degree in computer science and mathematics from Bloomsburg University in 1987 and the MSc and PhD degrees from the University of Maryland, College Park, in 1989 and 1993, respectively. Since 1993, he has been a codirector of the Laboratory for Language and Media Processing in the Institute for Advanced Computer Studies at the University of Maryland and an adjunct member of the graduate faculty. His

team of 15-20 researchers focuses on topics related to document image analysis and multimedia information processing. The team's recent intelligent document image analysis projects include page decomposition, structural analysis and classification, page segmentation, logo recognition, document image compression, duplicate document image detection, image-based retrieval, character recognition, generation of synthetic OCR data, and signature verification. In video processing, projects have centered on the segmentation of compressed domain video sequences, structural representation and classification of video, detection of reformatted video sequences, and the performance evaluation of automated video analysis algorithms. He is a coditor of the *International Journal on Document Analysis and Recognition*. He has more than 25 journal publications and almost 100 refereed conference proceedings. He is a member of the IEEE.



**Stefan Jaeger** received the bachelor's degree in computer science from the University of Kaiserslautern, Germany, in 1994 and the PhD degree in computer science from Albert-Ludwigs University, Freiburg, Germany, in 1998. His PhD dissertation addressed the problem of recovering dynamic information from static handwritten word images, for which he received the Dissertation Prize from the German Research Centers for Artificial Intelligence in 1999. From 1994 to 1998,

he was with the Daimler-Benz Research Center, Ulm, Germany, where he was engaged in offline cursive handwriting recognition for postal mail sorting. In 1998, he joined the Interactive Systems Laboratories jointly located at Carnegie Mellon University, Pittsburgh, and the University of Karlsruhe, Germany, where he was a research staff member and was responsible for online handwriting recognition and pen computing. From 2000 to 2003, he was a senior researcher in the Department of Computer, Information, and Communication Sciences, Tokyo University of Agriculture and Technology. From 2003 to 2007, he was with the Laboratory for Language and Media Processing in the Institute for Advanced Computer Studies at the University of Maryland. Since September 2007, he has been with the CAS-MPG Partner Institute of Computational Biology, which was jointly established by the Chinese Academy of Sciences (CAS) and the German Max-Planck Society (MPG). His research interests include computational biology, pattern recognition, machine learning, artificial intelligence, and sensor fusion.