Script-Independent Text Line Segmentation in Freestyle Handwritten Documents

LI Yi[1*], Yefeng Zheng[2], David Doermann[1], and Stefan Jaeger[1]

[1]Language and Media Processing Laboratory
Institute for Advanced Computer Studies
University of Maryland, College Park, MD 20742
{liyi,doermann,jaeger}@cfar.umd.edu

[2]Siemens Corporate Research
755 College Road East, Princeton, NJ 08540
E-mail: yefeng.zheng@siemens.com

*Author for correspondence
Phone: 1-301-405-3866
Fax: 1-301-314-2644

Submitted to IEEE Transactions on Pattern Analysis and Machine Intelligence as a regular paper.
Note: This paper contains color figures.

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]–[3]. Further experiments show 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 in 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 piece-wise 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.

Figure 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, though 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

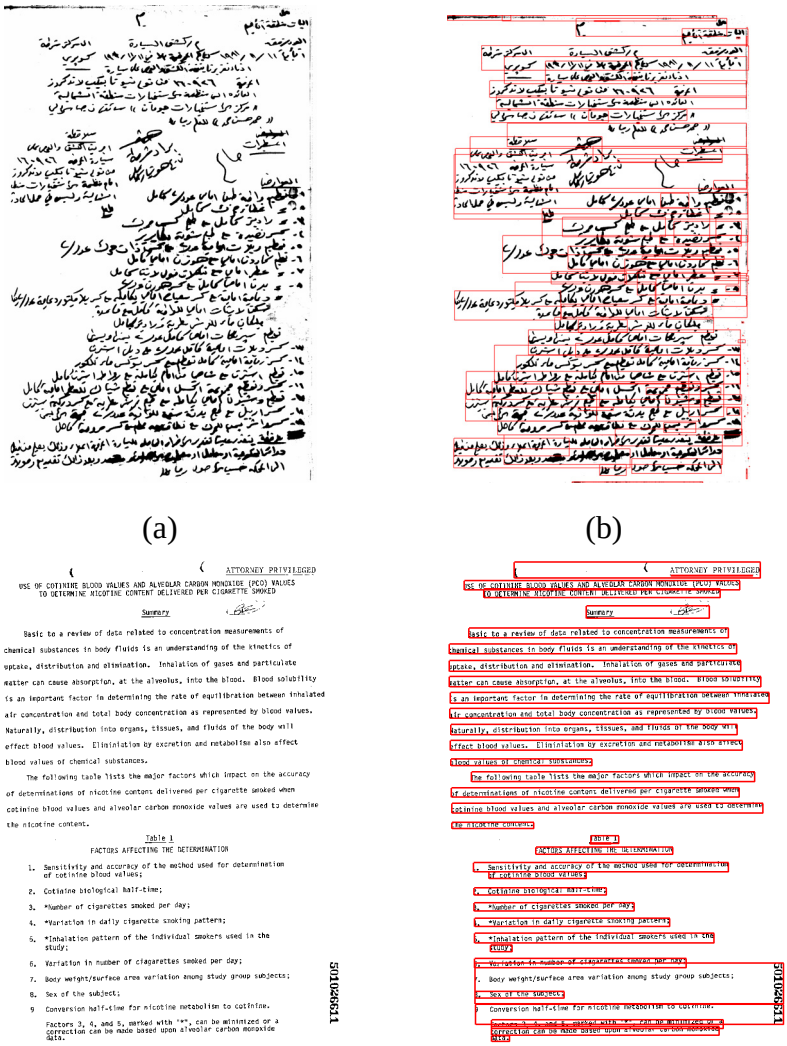


Fig. 1. (a) A handwritten document image; (c) A machine printed document image. The text line segmentation results using the Docstrum method [1] are shown in (b) and (d) using bounding boxes, respectively.

zones, text lines, words, and characters. However, it is not trivial to use bounding boxes to label overlapping curvilinear text lines, as shown in Figure 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, work directly on the input image (generally, a binary image). For connected component analysis, each pixel is treated equally 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 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 Figure 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 distribution function (PDF) of two dimensional variables that represents the text line distribution.
2) With the 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, i.e., script independence of the projection based methods and capability of processing a complex layout of the connected component based methods.

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

This paper has the following organization. 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 Section 3 and Section 4, respectively. Section 5 presents an extensive performance evaluation and quantitative comparison of the proposed method and the previous methods on a large handwriting dataset. This paper ends with a summary and a brief discussion of 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 used frequently in previous work for layout analysis [10]. In the bottom-up methodology, connected components are grouped into lines, and lines are grouped into zones, and so on. In the top-down methodology, a document page is first segmented into zones, and 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 freestyle handwritten document due to 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 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 be roughly categorized as bottom-up or top-down. As a bottom-up approach, connected component based

methods [2], [11] merge neighboring connected components 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 of 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 deskewing [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 used directly 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, 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 (e.g., in Chinese) or cross the rule lines (e.g., 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 a writer, they degrade the performance. For example, rule lines

make the 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 non-parametric method used in our approach. Without loss of generality, we assume character pixels are black and represented as "1"s in a binary image, while background pixels are white and "0"s. We also assume 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, e.g., up to $\pm 10^{\circ}$, is still allowed. However, zone segmentation [19] may be necessary to separate text lines with different orientations (e.g., horizontal versus vertical) into different zones if they appear on the same document.

3.1 Motivation

Performing segmentation directly on a binary image has been accepted without criticism for decades, although it happens that some difficulties are inherently involved. No evidence exists that humans use solely 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 better algorithms. A certain pattern naturally occurs when human beings write text lines in documents. For example, 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 probability density function that generates this pattern. Instead of segmenting directly 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 (Figure 2b). This estimated probability map offers many working advantages. We can avoid calculating some geometric properties, such as centroid and connected component size, at 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 probability density function (PDF). This PDF represents the distribution of text lines. Specifically, the PDF is continuous and has larger values in the text line area, while 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 indicate they 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 multi-modal densities, whose density functions are usually unknown. Therefore, non-parametric density estimation should be used. The Parzen window [21] is one non-parametric density estimation technique. Isotropic kernels are frequently used if no domain-specific prior is given [22]. Without loss of generality, we assume the reading order of text line 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 $x = 1, \ldots, M$ and $y = 1, \ldots, N$, we want to estimate the probability map $PDF_{est}(x, y)$, for $1 \leq x \leq M$ and $1 \leq y \leq M$, from the observation $I(x, y)$. Kernel-based non-parametric methods are widely used in statistics. Given a continuous Gaussian kernel φ in a two dimensional 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 σ_x and σ_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 Figure 2b) on the text lines, but low density values (light) for the gaps.

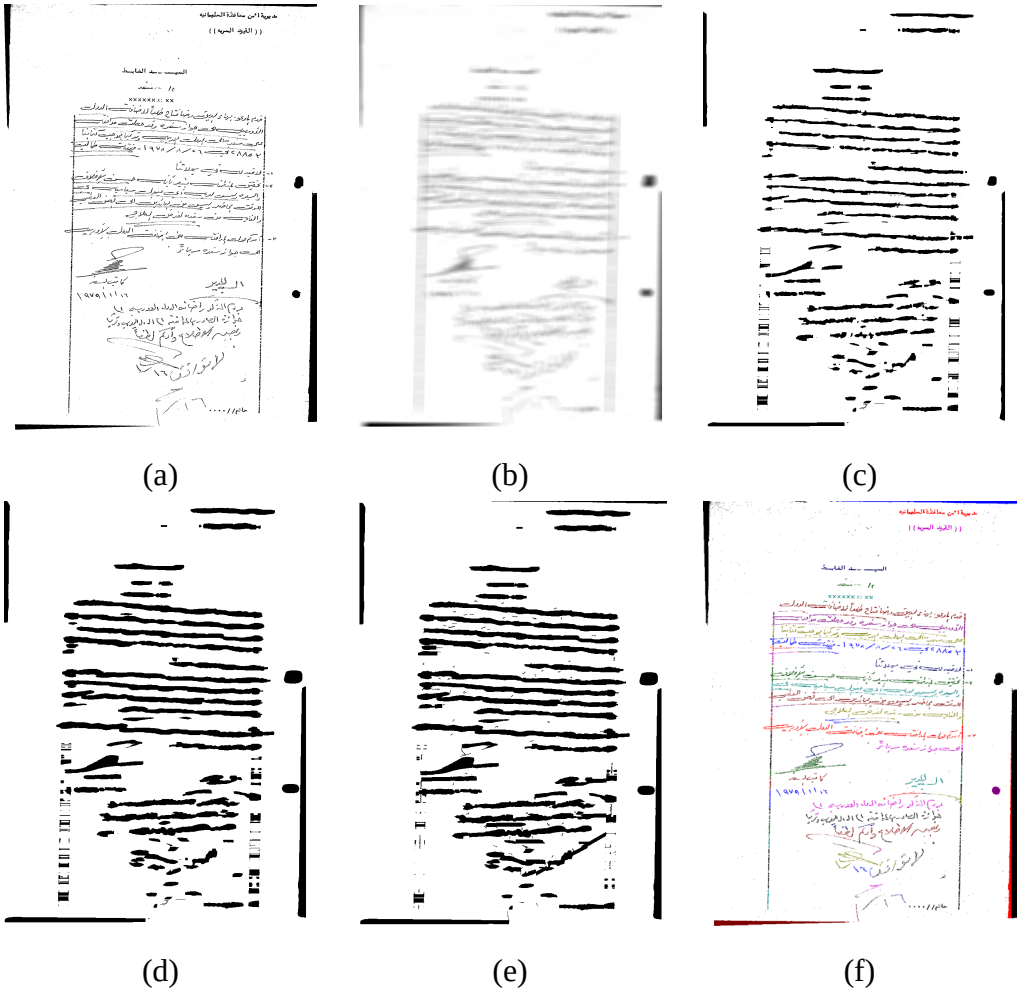


Fig. 2. Illustration of the proposed text line segmentation method. (a) A handwritten document; (b) Density estimation using anisotropic kernel; (c) Initial estimate of text lines; (d) Result after 10 iterations of evolution using the level set method; (e) After connecting a few broken text lines and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

Parzen window techniques guarantee the result PDF is continuous and smooth if 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 Figure 2b, the details 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 indicate they are between neighboring text lines boundaries, as shown in Figure 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 Figure 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 theory. 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 grow and merge automatically. It has been adopted successfully in many fields, such as fluid mechanics, computer animation, and computer vision. The level set method deals with topological changes naturally 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].

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

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 usually is 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 with 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 (Equation (2)). So the zero level set grows, merges, and stops automatically to the final text line

boundary, guided by the PDE. The numerical solution of Equation (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 (Figure 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 (e.g., 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 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 Equation (2) is set to the square of actual curvature. By our assumption, text lines are horizontally elongated, therefore, curvatures of left and right ends of boundaries are larger than those of top and bottom, which means 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. So, PDF_{est} actually controls the speed of the boundary.

Another constraint 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.

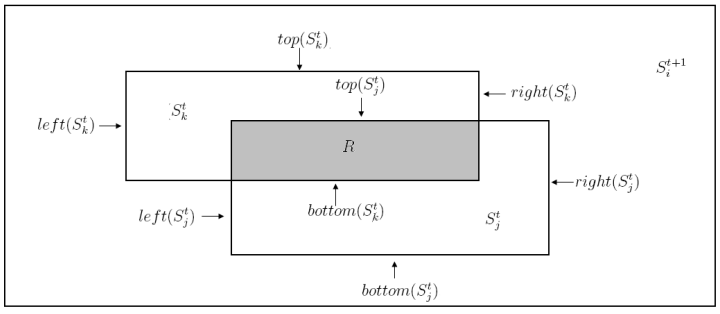
At time t we retrieve the zero level set, and we obtain N bounding boxes of closed region $S_n^t, n = 1, \ldots, N$. Suppose, after iteration $t+1$, the zero level sets are $S_n^{t+1}, n = 1, \ldots, M, M \leq N$. For each S_n^{t+1}, it must contain one or more regions of S_n^t's since all regions are growing. Assume 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:

$$left(S_k^t) \leq mean(S_j^t) \leq right(S_k^t) \tag{3}$$

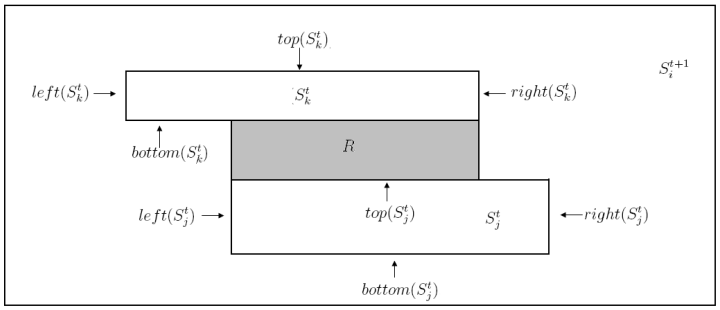
or

$$left(S_j^t) \leq mean(S_k^t) \leq right(S_j^t) \tag{4}$$

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



(a)



(b)

Fig. 3. The illustration of the collision.

If Equation (3) or Equation (4) holds, a collision happens. We then calculate the collision region, R, as follows. Without loss of generality, we assume S_j^t is under S_k^t in the image, but

two regions do not necessarily intersect (Figures 3a and 3b). That is, $top(S_k^t) \leq top(S_j^t)$ given the origin is the upper left corner of the image. We define the boundary of region R as the follows:

$$top(R) = min(bottom(S_k^t), top(S_j^t)) \tag{5}$$

$$bottom(R) = max(bottom(S_k^t), top(S_j^t)) \tag{6}$$

$$left(R) = max(left(S_k^t), left(S_j^t)) \tag{7}$$

$$right(R) = min(right(S_k^t), right(S_j^t)) \tag{8}$$

If collisions occur, we roll back to time t, and 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 $LS_{initial}$ using the Niblack algorithm [24].
2) Initialize the level set function f using the initial estimate $LS_{initial}$. A simple initialization procedure calculates the distance transform of $LS_{initial}$ as follows:

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

   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 (Equation (2)).
4) Vertical merge detection, as described above.
5) Repeat Step 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 10 iterations are usually enough to achieve good results. For details of the level set method, please refer to [23].

4.3 Post-Processing

After growing the initial estimate of text lines using the level set method, most text line boundaries are detected correctly, but a few fragments may still exist (Figure 2d) due to extraordinary large horizontal word gaps. These fragmented pieces can be easily linked to obtain the final text lines using a rule-based approach similar to [1].

We briefly describe our rule as follows.

1) Segments are assigned to unique IDs.
2) Calculate the orientation of each line segment 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, extend the line to left or right using following merging rules: we merge it to another "unprocessed" line segments if a) the difference of their orientations is less than 10°, b) their bounding boxes' horizontal gap is less than $\frac{1}{10}$ of the page length, and c) their bounding boxes overlaps more than 50% in the vertical direction. The ID of the merged segment is then updated. We apply this rule iteratively until no more "unprocessed" line segments in left or right satisfying the merging conditions.
6) Mark all segments involved in step 4 as "processed."
7) Got 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 isolate connected components and the closest text line is less than the estimated text line height, we label these isolate 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 (e.g. diacritics). Figure 2e shows the result after broken

line segment linking and grouping the isolated connected components. Figure 2f shows the final text line segmentation result 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 I provides statistics about the number of component 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 of the connected components, on average. The number of components also drops significantly to approximately 1/10 of the original number. We further reduce the number of components by grouping broken line segments and isolated small pieces.

TABLE I

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

5 EXPERIMENTS

5.1 Dataset

In this section, we present a quantitative analysis of our algorithm on a large database. The UMD test dataset was collected by the Language and Media Processing Laboratory at the University of Maryland[1]. Hard copies of handwritten documents were collected from many countries, such as China, India, Korea, and Japan, and scanned at a resolution of 300 dot-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 II shows some statistics for each script. A typical image size is approximately 2000×1500 pixels, and the character sizes vary from roughly 10×10

[1]The dataset and the elevation algorithms are available for direct comparison by contacting the authors.

to 30×30 pixels. We also collected a large dataset of Arabic documents containing 166,071 images. We have selected some for our experiments. Figure 4 shows some examples.

TABLE II

SOME STATISTICS OF THE UMD DATASET.

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

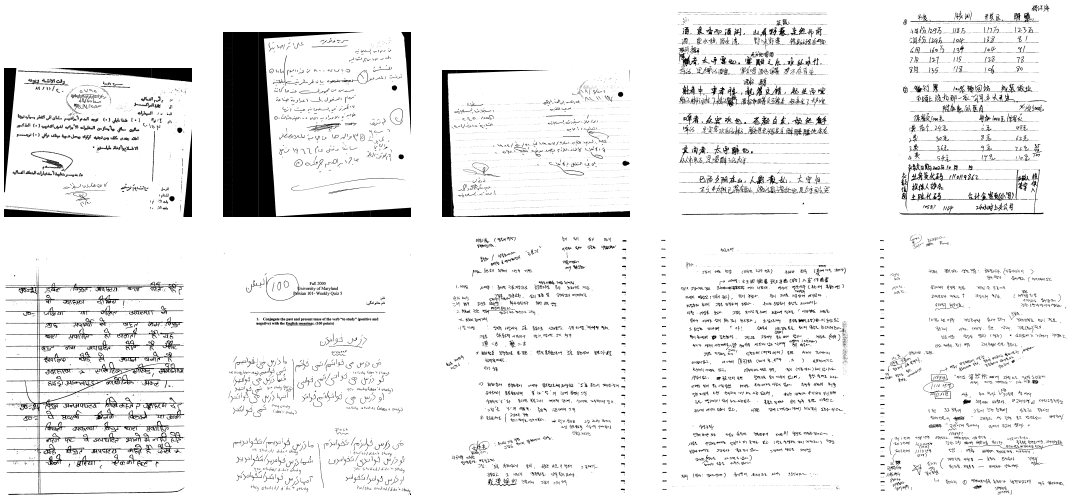
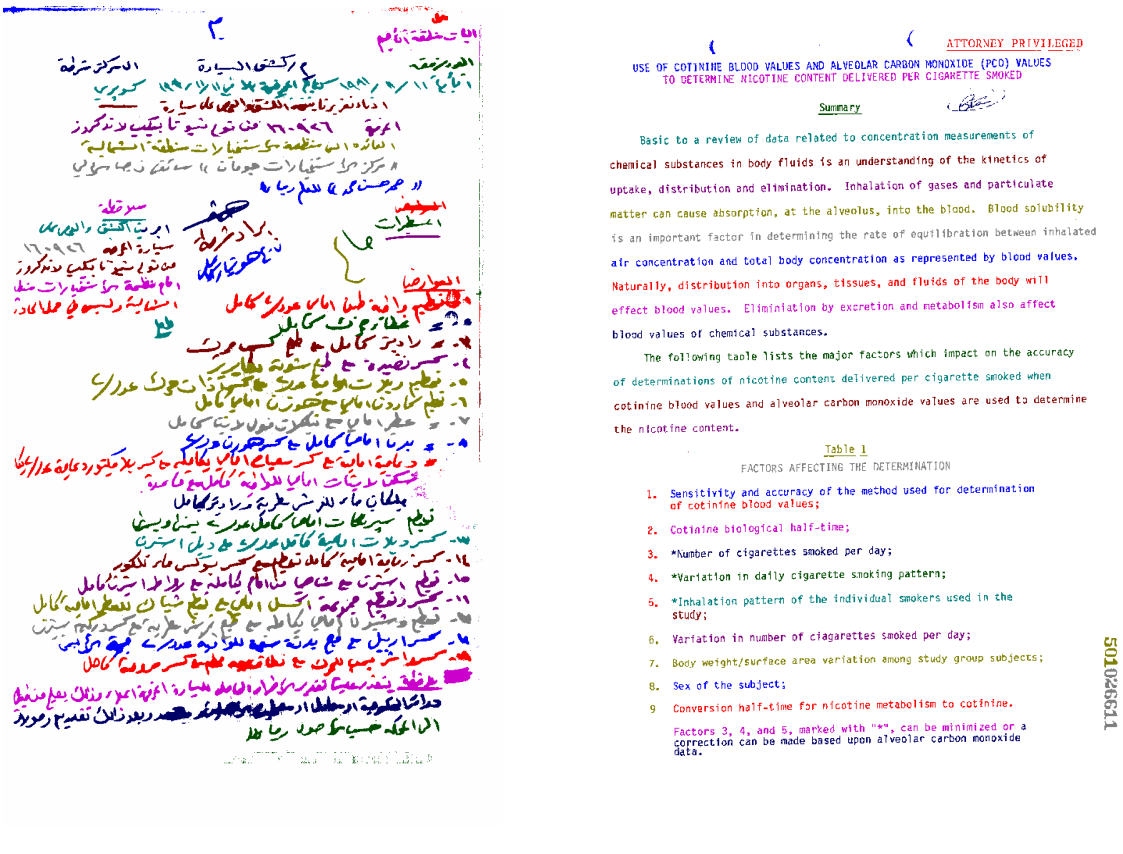


Fig. 4. Handwriting examples in the UMD dataset.

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, 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, and test our algorithm on these synthesized images. Segmented results are visualized using different colors, which denote different text lines. Figure 5 shows the segmentation results of our algorithm on Figures 1a and 1c. More examples appear in Figure 6.



ATTORNEY PRIVILEGED

USE OF COTININE BLOOD VALUES AND ALVEOLAR CARBON MONOXIDE (PCO) VALUES TO DETERMINE NICOTINE CONTENT DELIVERED PER CIGARETTE SMOKED

Summary

Basic to a review of data related to concentration measurements of chemical substances in body fluids is an understanding of the kinetics of uptake, distribution and elimination. Inhalation of gases and particulate matter can cause absorption, at the alveolus, into the blood. Blood solubility is an important factor in determining the rate of equilibration between inhalated air concentration and total body concentration as represented by blood values. Naturally, distribution into organs, tissues, and fluids of the body will effect blood values. Elimination by excretion and metabolism also affect blood values of chemical substances.

The following taole lists the major factors which impact on the accuracy of determinations of nicotine content delivered per cigarette smoked when cotinine blood values and alveolar carbon monoxide values are used to determine the nicotine content.

Table 1

FACTORS AFFECTING THE DETERMINATION

1. Sensitivity and accuracy of the method used for determination of cotinine blood values;
2. Cotinine biological half-time;
3. *Number of cigarettes smoked per day;
4. *Variation in daily cigarette smoking pattern;
5. *Inhalation pattern of the individual smokers used in the study;
6. Variation in number of ciagarettes smoked per day;
7. Body weight/surface area variation among study group subjects;
8. Sex of the subject;
9. 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.

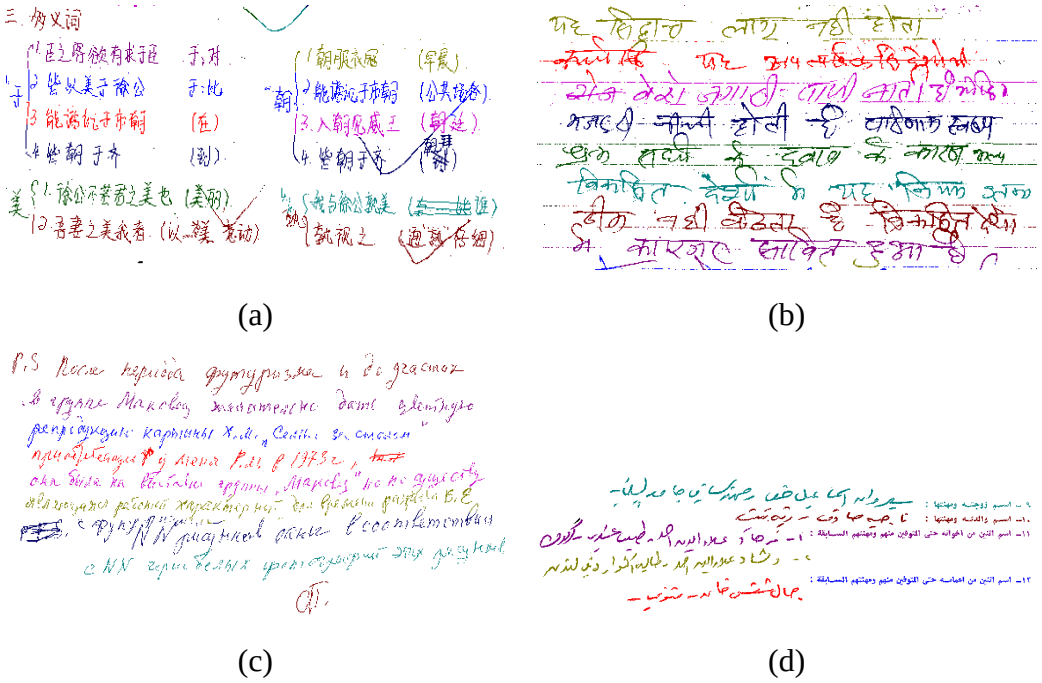
501026611

(a) (b)

Fig. 5. Our algorithm's results for Figures 1a and 1c, respectively.

In all these experiments, we use the same parameters. Specifically, we use the same Gaussian window and standard deviation for density estimation, and the number of iteration 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 30×120 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 dataset.

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 using the VIPER Ground Truth Editor [26]. Since bounding boxes are not sufficiently accurate to delineate the boundary of text lines, we use polygons for ground-truthing. In total, there are 7,660 handwritten text lines.



(a) (b) (c) (d)

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) A document mixed with handwritten and machine printed 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 four coordinates of a bounding box. However, some overlaps inevitably occur among bounding boxes in curvilinear handwritten documents. Non-overlapping 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.

Suppose there are M ground-truthed lines and N detected lines, we construct an $M \times N$ matrix P. An element $P_{i,j}$, for $i = 1, \ldots, M$ and $j = 1, \ldots, N$, of matrix P represents the number of shared black pixels between the i^{th} ground-truthed text line and 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 $S(k)$, $k = 1, \ldots, max(M, N)$, the goodness $G(S)$ of this assignment is the total number of shared black pixels.

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

The best assignment S_o is the one with maximum goodness

$$S_o = arg \max_S G(S). \tag{11}$$

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

The overall pixel-level hit rate is defined as

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

By using the pixel-level hit rate and the Hungarian algorithm, different segmentation errors, e.g., splitting, merging, and missing, can be appropriately penalized with weights 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

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

and,

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

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

5.4 Performance and Comparison

We organize our discussion on performance and comparison as follows. First, we briefly show the accuracy on four scripts, then we discuss the performance on two different categories, handwriting with and without rule lines. After that, we compare our performance with three other techniques. As a top-down method, X-Y cut [3] builds a structural tree that represents the document layout. The projection histogram analysis determines the cut in each step. Docstrum

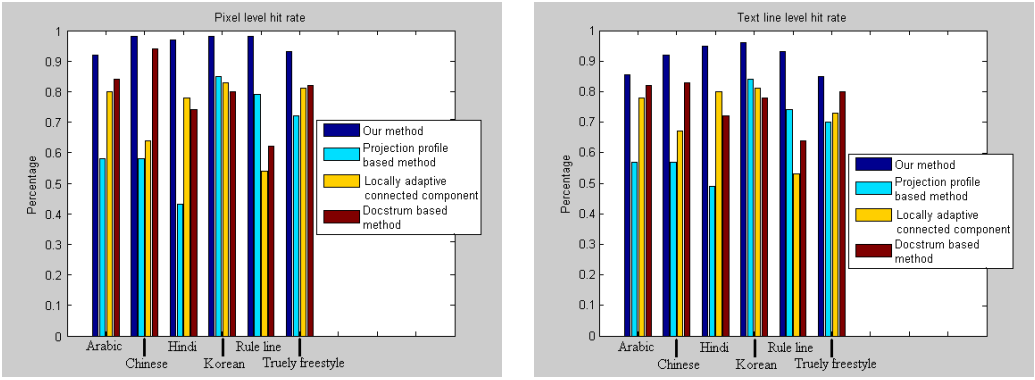
[1], on the other hand, builds the layout structure bottom-up from the connected components. We can, therefore, retrieve the text line level from the tree structure, although these methods are not designed solely for text line segmentation. Our comparison experiments use a public domain implementation [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 addressed sufficiently in previous work, we can only compare our approach to those methods that are originally designed for machine printed documents.

In principle, we can train the optimal parameters of an algorithm based on ground truth as Mao *et al*. did in [28] for the evaluation of different page segmentation algorithms. As shown in Mao's 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 fix these parameters for all the experiments.

5.4.1 Performance of Our Algorithm

Our method's performance for different scripts is shown in the Figure 7, with details listed on the second row of Tables III - VI. Overall, we achieve 94.7% accuracy in terms of pixel-level hit rate. The standard deviation of the accuracy on all scripts is small, which shows our algorithm is stable. At the text line level, there are a total of 7,660 ground-truthed lines in Tables III-VI, and our approach can detect 6,982 (91.2%) of them correctly. Details are shown in the last column of Tables III-VI. Our algorithm does not use script-dependent information, and we set the same parameter for all scripts. The slightly degradation in performance is observed on Arabic documents because they are less constrained (e.g., there might not exist consistent base lines for handwritten Arabic text lines (Figure 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 (e.g., Figure 8a). Many, such as lecture notes and some drafts, are written on papers with rule lines (e.g., Figure 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



(a) (b)

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

TABLE III

QUANTITATIVE COMPARISON ON 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%)

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 row of Tables VIII and VII, respectively. At the text line level, our approach can detect 1,466 out of 1,589 (93%) for documents with rule lines, and 1,863 out of 2,178 (85%) for freestyle documents.

Generally, the major failures happen because two neighboring text lines touch each other significantly (e.g., Figure 9). Since two lines are connected in only few touched areas, a post-processing to segment them is necessary to improve performance. We also notice the correction in

TABLE IV

QUANTITATIVE COMPARISON ON 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%)

TABLE V

QUANTITATIVE COMPARISON ON 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%)

the gap between two lines (Figure 9a) may create many peaks in the density function estimation, and it is not trivial to segment them. Other challenges include non-text objects such as stamps, logos, and severe noise. We will measure the affect 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 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 Figure 7, with details listed on the third through fifth rows of Tables III-VI. We achieve much better results in comparison to other methods, regardless of scripts used. For example, the pixel-level hit rate of the connected component based method is 78% in the Hindi dataset (Table V), while our method is 97%. At

TABLE VI

QUANTITATIVE COMPARISON ON 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%)

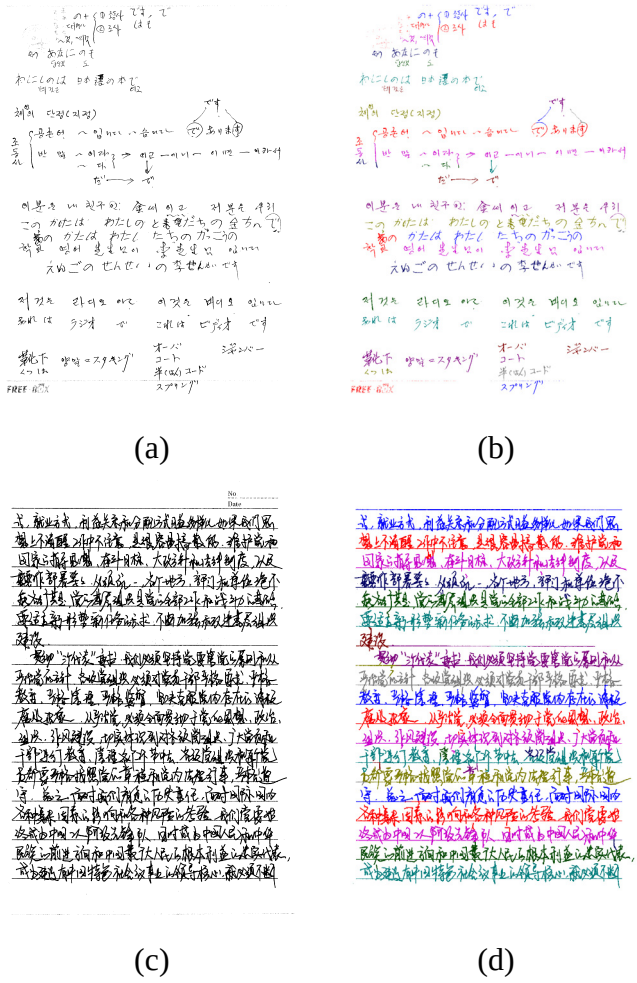
TABLE VII

QUANTITATIVE COMPARISON ON 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%)

the text line level, these methods performs significantly worse. Table V shows only 996 (43%) text lines are detected correctly using projection profile method in Hindi documents, which is significantly worse than our method (95%). This is primarily because the top line of the Hindi words and the rule lines introduce many false peaks in projection profile analysis. Generally, our algorithm is quite stable for all four scripts.

Table VIII 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 VII). Comparing the same algorithm in two categories, we observe 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 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



(a) (b)

(c) (d)

Fig. 8. A freestyle document image (a) and a document image with rule lines (c). Results of our algorithm are shown in (b) and (d), respectively

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 Figures 10c and 10d. Text lines are often broken or merged incorrectly. In our result (e.g., Figure 10a), most text lines are clearly labeled without confusion.

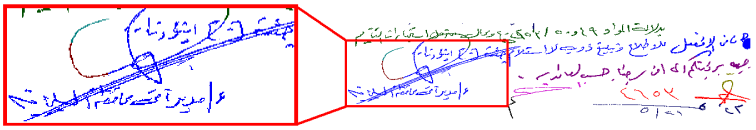
A good text line segmentation algorithm should process machine printed documents as well. Figure 5b shows the segmentation results on a machine printed document without tuning parameters. It shows our algorithm performs better or comparably to traditional methods on machine

TABLE VIII

QUANTITATIVE COMPARISON ON 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%)

(a)

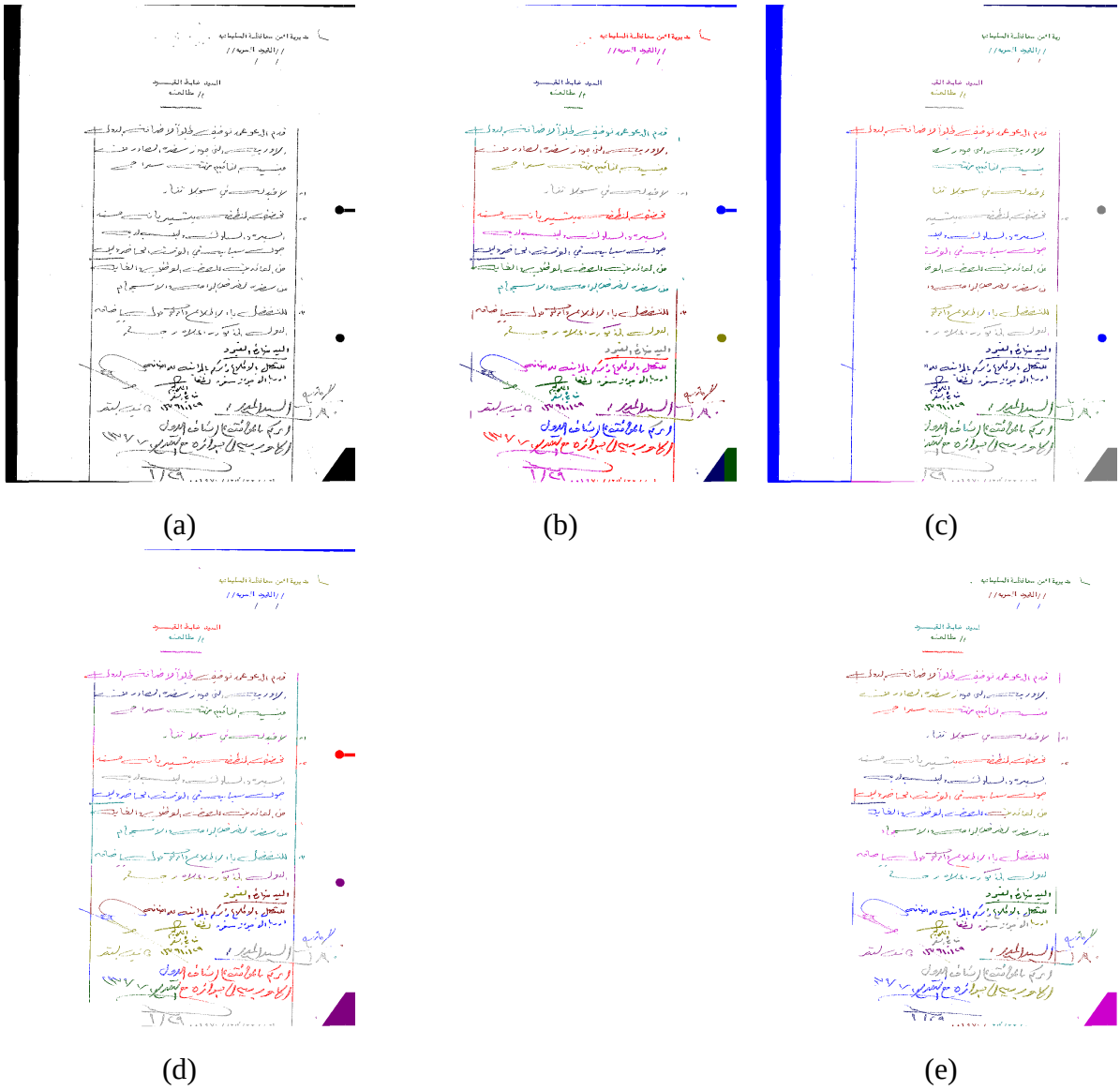


(b)

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

printed documents.

The connected component based methods 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 our method can be used for handwritten documents, machine printed documents, and documents mixing with machine-printed text and handwriting. Our assumption fits well for both machine-printed and handwritten text lines. The statistical results show our method is better in handwritten documents than classical methods and their variations.



(a) (b) (c)

(d) (e)

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

5.5 Robustness Test

In this section we examine the robustness of our algorithm to both changes in internal parameters and changes to 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.

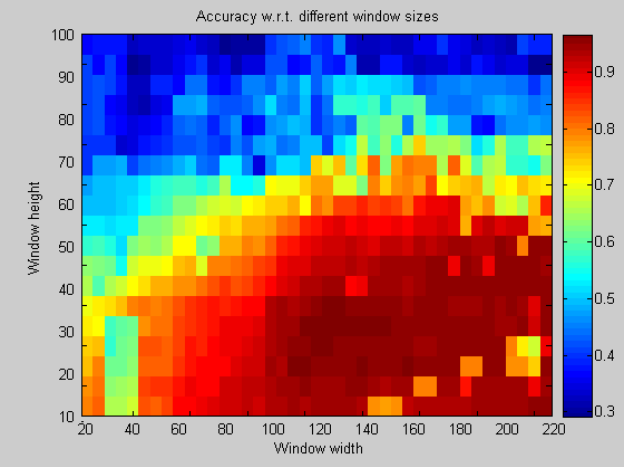


Fig. 11. Pixel-level hit rate of our method under the change of window size.

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 reasonable number of iterations. In experiments, we did not find a significant difference even we increase the iteration number from 10 to 50. We, therefore, evaluate the Parzen window size in following.

We change the window size while keeping the standard deviations proportional to the window size. In this way, we evaluate the performance of internal parameters. Figure 11 shows the performance of different Parzen windows. The accuracy is visualized using a color map. The x axis is the width of kernel window, the y axis is the height of kernel window. The best size of the kernel window is 30×120 in our case. In all 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 of scale and rotation on the four annotated datasets. The character size within images is affected by 1) scanning resolution and 2) font size. Therefore, we can evaluate the sensitivity of our method by resizing the image to mimic the change of character size. Figure 12 shows accuracy for different scales. It shows

our algorithm produces similar results with different character sizes. One example appears in Figure 13.

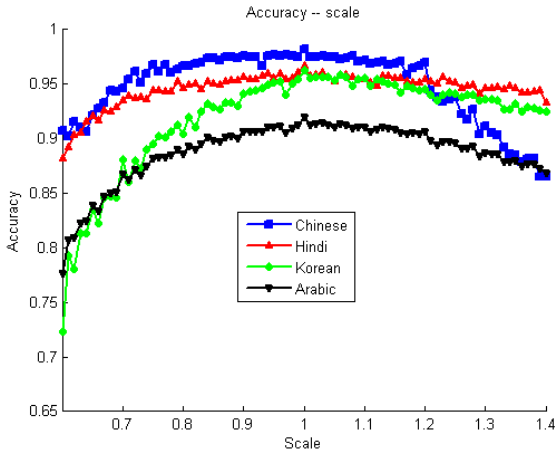


Fig. 12. Pixel-level hit rate of our method under the change of scale.

To illustrate how the skew degrades the performance, we rotate an image using different angles, and compare the accuracy. Figure 14 shows the quantitative evaluation under different rotation angles and Figure 15 shows one example under three different rotation angles. The performance of our algorithm does not degrade significantly when orientation changes within [-10°, 10°]. For most deskewed handwritten documents, the skew falls into this range. Experiments show our algorithm can fit in most cases, with a deskew module as preprocessing.

By using the best internal parameters from Section 5.5.1, we observe a small peak in Figures 12 and 14. Overall, our performance shows 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 (e.g., $\pm$ 10°).

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 the 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.

(a) (b) (c) (d)

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%; (d) Result on the image scaled to 120%.

In this experiment, we demonstrate the performance using salt-and-pepper noise. By randomly flipping the pixels in binary images, we can perform simulations with different probability. Figure 17 shows three images with different noise levels. Figure 16 gives a statistical analysis of accuracy in terms of different noise probability for four scripts.

Results show our algorithm degrades elegantly under noise, and noise reduction and text restoration [29]–[31] are helpful but 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 2000×1500 pixels nears 20 seconds on a PC with 1.6 GHZ CPU and 1 GB 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.

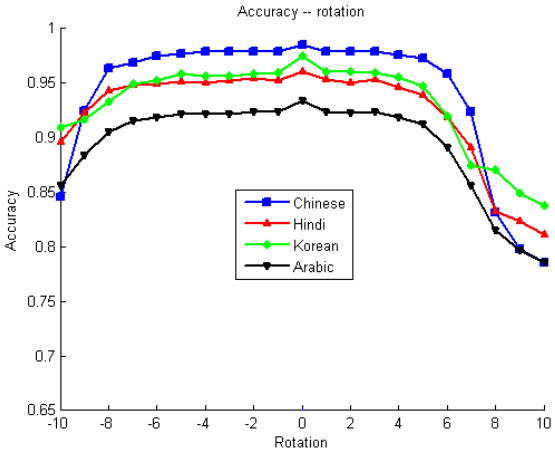


Fig. 14. Performance of our method under the change of rotation angles.

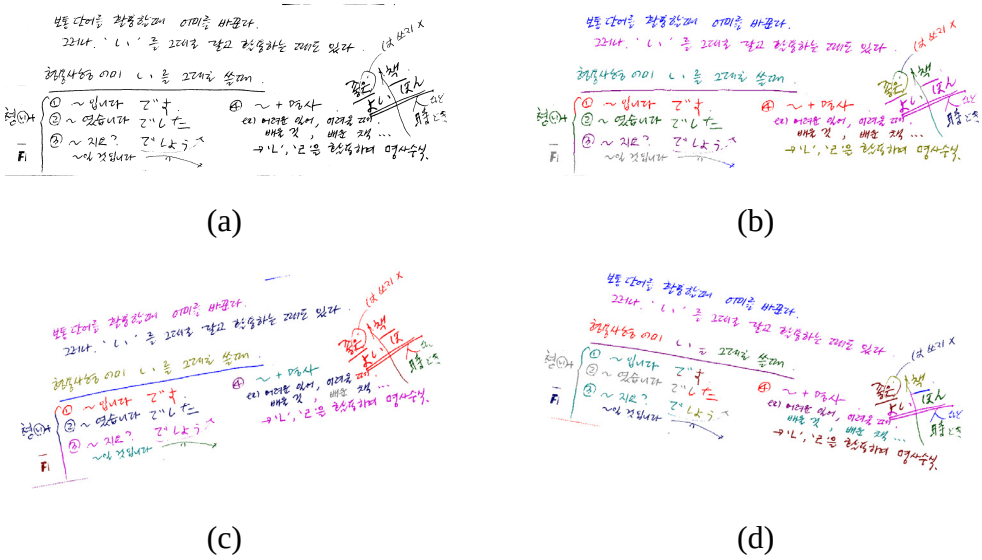


Fig. 15. Segmentation results for documents with different rotations. (a) Original image; (b) Result on the original image; (c) Result on the image rotated -10°; (d) Result on the image rotated 10°.

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 the 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

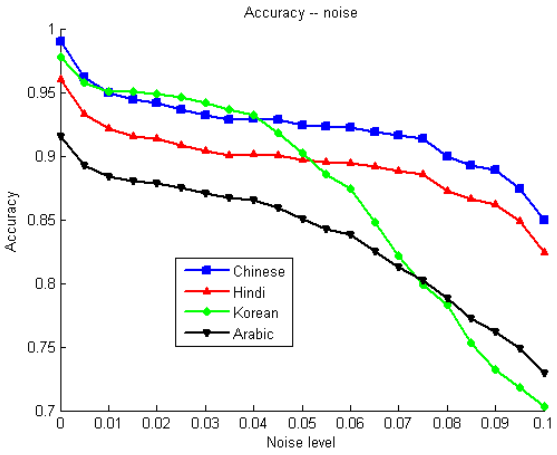
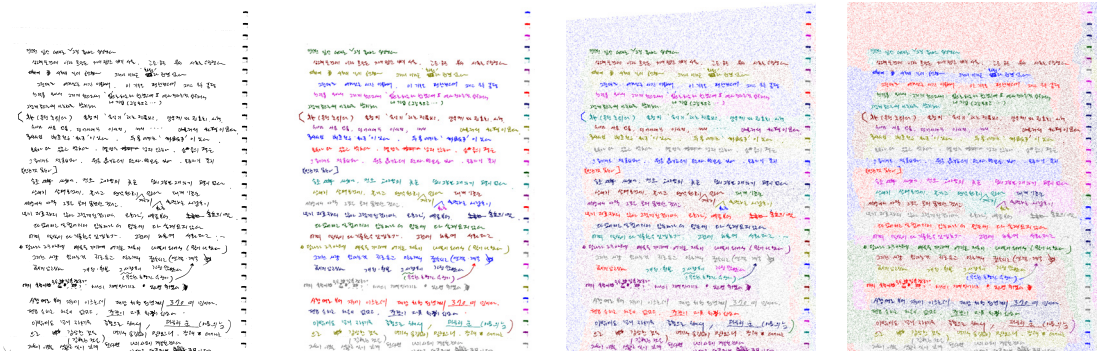


Fig. 16. Performance of our method under the change of noise levels.



(a) (b) (c) (d)

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% noise; (d) Result on the image with 10% noise.

consistently high accuracy on many scripts, such as Arabic, Hindi, Chinese, and Korean, using the same parameter setting. Experiments show our algorithm is robust, and it degrades elegantly under the changes of scales, rotation, and noise. Though originally developed with a binary image in mind, our algorithm can extend to gray-scale and color document images without major change. 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 of 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 as priori, 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 many top-down methods. We consider pixels instead of characters in our approach, and estimate the unknown probability density function 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 post-processing 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 anonymous reviewers for the constructive comments, and the support of this research by the US Department of Defense under contract MDA-9040-2C-0406 is gratefully acknowledged.

REFERENCES

[1] L. O'Gorman, "The document spectrum for page layout analysis," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 15, no. 11, pp. 1162–1173, 1993.

[2] S. Jaeger, G. Zhu, D. Doermann, K. Chen, and S. Sampat, "DOCLIB: A software library for document processing," in *Document Recognition and Retrieval XIII, Proc. of SPIE vol. 6067*, 2006, pp. 63–71.

[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, 1992.

[4] R. Plamondon and S.N. Srihari, "Online and offline handwriting recognition: A comprehensive survey," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 22, no. 1, pp. 63–84, 2000.

[5] Y. Li, Y. Zheng, and D. Doermann, "Detecting text lines in handwritten documents," in *Proc. Int'l Conf. Pattern Recognition*, 2006, pp. 1030–1033.

[6] Y. Li, Y. Zheng, D. Doermann, and S. Jaeger, "A new algorithm for detecting text line in handwritten documents," in *Int'l Workshop on Frontiers in Handwriting Recognition*, 2006, pp. 35–40.

[7] S. Mao, A. Rosenfeld, and T. Kanungo, "Document structure analysis algorithms: a literature survey," in *Document Recognition and Retrieval X, Proc. of SPIE vol. 5010*, 2003, pp. 197–207.

[8] A.K. Jain and B. Yu, "Document representation and its application to page decomposition," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 20, no. 3, pp. 294–308, 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 Anal. Machine Intell.*, vol. 22, no. 1, pp. 38–62, 2000.

[11] A. Simon, J.-C. Pret, and A. Johnson, "A fast algorithm for bottom-up document layout analysis," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 19, no. 3, pp. 273–277, 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 Anal. Machine Intell.*, vol. 27, no. 8, pp. 1212–1225, 2005.

[14] U.-V. Martin and H. Bunke, "Text line segmentation and word recognition in a system for general writer independent handwriting recognition," in *Proc. Int'l Conf. Document Analysis and Recognition*, 2001, pp. 159–163.

[15] N. Tripathy and U. Pal, "Handwriting segmentation of unconstrained Oriya text," in *Int'l Workshop on Frontiers in Handwriting Recognition*, 2004, pp. 306–311.

[16] U. Pal and P.P. Roy, "Multioriented and curved text lines extraction from Indian documents," *IEEE Trans. System, 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 hand-written text-line extraction," in *Proc. Int'l Conf. Document Analysis and Recognition*, 2001, pp. 281–285.

[18] Y. Zheng, H. Li, and D. Doermann, "A parallel-line detection algorithm based on HMM decoding," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 27, no. 5, pp. 777–792, 2005.

[19] J. Sauvola, D. Doermann, and M. Pietikainen, "Locally Adaptive Document Skew Detection," in *Document Recognition and Retrieval IV, Proc. of SPIE vol. 3027*, 1997, pp. 96–108.

[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," *Ann. Math. Stat.*, vol. 33, no. 3, pp. 1065–1076, 1962.

[22] R.O. Duda, P.E. Hart, and D.G. Stork, *Pattern Classification*, 2nd ed. Wiley-Interscience Publication, 2000, pp. 84–97.

[23] B. Sumengen, "Variational image segmentation and curve evolution on natural images," Ph.D. dissertation, University of California, Santa Barbara, Sept. 2004.

[24] W. Niblack, *An Introduction to Digital Image Processing*. Prentice Hall, Englewood Cliffs, NJ, 1986, pp. 115–116.

[25] A. Rosenfeld and J. Pfaltz, "Sequential operations in digital picture processing," *J. of Assoc. Compt Machinary*, vol. 13, no. 4, pp. 471–494, 1966.

[26] D. Doermann and D. Mihalcik, "Tools and techniques for video performances evaluation," in *Proc. Int'l Conf. Pattern Recognition*, 2000, pp. 167–170.

[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," in *Proc. Symp. Document Analysis and Information Retrieval*, 1993, pp. 1–16.

[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, no. 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 University Press, 1999.