A New Algorithm for Detecting Text Line in Handwritten Documents

Yi Li, Yefeng Zheng, David Doermann, Stefan Jaeger

A New Algorithm for Detecting Text Line in Handwritten Documents

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

[1]Laboratory for Language and Media Processing
Institute for Advanced Computer Studies
University of Maryland, College Park, MD 20742-3275
E-mail: {liyi, doermann, jaeger}@umiacs.umd.edu

[2]Siemens Corporate Research
755 College Road East, Princeton, NJ 08536
E-mail: yefeng.zheng@siemens.com

Abstract

Curvilinear text line detection and segmentation in handwritten documents is a significant challenge for handwriting recognition. Given no prior knowledge of script, we model text line detection as an image segmentation problem by enhancing text line structure using a Gaussian window, and adopting the level set method to evolve text line boundaries. Experiments show that the proposed method achieves high accuracy for detecting text lines in both handwritten and machine printed documents with many scripts.

Keywords: Handwritten Text Line Detection, Document Image Analysis, Level Set Methods

1 Introduction

Text lines in handwritten documents are often curvilinear and close to each other (Fig. 1a). In most work on handwritten document images, the authors either assume that text lines are segmented, or that the gap between two neighboring text lines is large enough so that conventional methods can be applied [1, 2]. However, it is not easy to extend algorithms for machine printed documents to handwritten documents where text lines are not perfectly straight. Therefore, handwritten text line segmentation is still a major challenge in document image analysis. It is important because it provides information for skew correction, zone segmentation, and character recognition. In this section, we first review some previous work on text line detection in machine printed documents and the corresponding extensions for handwritten documents.

1.1 Related work

Previous work on text line segmentation can be roughly categorized as bottom-up or top-down. As a bottom-up approach, the connected component based methods [3] merge neighboring connected components using a few simple rules on the geometric relationship between neighboring blocks. We implemented an improved method based on connected component grouping [4], but we also found that these methods are sensitive to topological changes of the connected components, and it is not easy to derive script independent merging rules based on connected components. Projection based methods [5] may be one of the most successful top-down algorithms for machine printed documents. Because the gap between two neighboring text line is typically significant, the projection of text lines is easily separable in the orthogonal direction [6]. However, these methods cannot be used in handwritten documents directly, unless gaps between two neighboring handwritten lines are significant or handwritten lines are straight [2]. Another disadvantage of the top-down approaches is that they cannot easily process complex non-Manhattan layouts.

Almost all text line segmentation approaches for machine printed documents are based on the assumption that text lines are straight. Some approaches, e.g., the projection based methods, can be extended to deal with curved text lines. Tripathy et al. [7] attempts to divide the image into multiple columns, use a projection based method in each column, and combine the results of adjacent columns into a longer text line. Generally, a better result is achieved than with naive projection methods, but the merging of detected line segments of adjacent columns may be ambiguous so that it is still difficult to generate a reasonable result. Other researchers use different assumptions for their specific assumptions [8] or for a specific script [9], but our methods are more general to process heterogeneous handwritten documents.

1.2 Overview of Our Approach

We assume the text line orientations are roughly uniform, and a skew correction method can orient the text lines to be horizontal. A relatively large variation in orientation, e.g., up to 30°, is still allowed. However, zone segmentation may be necessary to separate text lines with different orientations into different zones if they appear

on the same document. One such method is presented in [10].

Text line segmentation is an image segmentation problem. Therefore, we model handwritten text lines in a general image segmentation framework. Unlike previous methods, we convert the original binary image to a gray-scale image to enhance the text line structure. Working on a smooth gray-scale image, we can use a state-of-the-art image segmentation approach, the level set method [11].

2 The Algorithm and Implementation

Experiments using distance transform [12] motivate us to transform binary image to gray scale so that the underlying structure can be retained. In our approach, we use Gaussian smoothing to convert a binary input to a gray-scale image. A rectangular window for blurring is adopted. The height of the window is smaller than the average height of the vertical gap between neighboring text lines to avoid blurring the gap. The width of the window can be relatively large, so the horizontal gap between neighboring words or characters is blurred. The resulting image IM_{avg} has a low pixel intensity (dark) on the text lines, but a high intensity (light) in the vertical gaps (as shown in Fig. 1b). By blurring the details of the handwritten strokes, the text line structures are enhanced, and we can achieve script independence. The resulting gray-scale image is smooth, therefore gradient-based image segmentation methods, e.g., the level set method, can be applied.

We simply binarize IM_{avg} to obtain the initial estimate of the text lines, $IM_{initial}$. The text lines we obtain here, however, may be fragmented (Fig. 1c). So, we need to grow these partial text lines to generate longer ones. Since the level set method is an excellent tool for boundary evolution, which deals with topological changes naturally, it is well-suited for our application.

2.1 Level Set Based Detection

The level set method [11] is an effective algorithm in image segmentation. It is based on a partial differential equation (PDE). The basic idea of the level set method is to evolve the boundary by its partial derivatives and an external vector field. A general formulation of the level set method is as follows [13]:

$$\frac{\partial f}{\partial t} + \vec{S} \cdot \nabla f + V_N |\nabla f| = b_k |\nabla f|, \qquad (1)$$

where $\frac{\partial f}{\partial t}$ denotes the move of the boundary frontier, which depends on the external vector field $\vec{S}$, gradient ∇f, speed V_N in the normal direction, and curvature b_k.

Our algorithm for evolving can be summarized as following:

1. Binarize IM_{avg} to obtain initial estimates $IM_{initial}$. The initial estimate need not to be very accurate, and as long as at least one part of a text line is extracted, we can achieve a good final segmentation result by growing regions.

2. Initialize the level set function f. 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 the zero value for the boundary. We use the initial estimate $IM_{initial}$ to initialize the level set function f. This is a standard process in the level set method [11].

3. Evolve the initial zero level set according to the PDE (Eq. 1). So it can grow, merge, and stop automatically in the complete text lines. In our experiments, IM_{avg} serves as the normal speed V_N, while the external vector field is set to be zero. Using IM_{avg} as the speed, the boundary will grow faster inside the text lines, where black pixel densities are large, while slower when it approaches to gaps. So, IM_{avg} actually controls the speed of the boundary.

In addition, with the priori knowledge that a text line is a horizontally elongated shape, we force the text line boundary to grow faster in the horizontal direction. Curvature is also adopted as another factor in evolving the boundary. With our assumption, text lines are horizontal, therefore curvatures of left and right ends of a text line are larger than those of top and bottom, which means that the 10 iterations to achieve good results. For details of the level set method, please refer to [13].

2.2 Text Line Localization

After growing the initial estimate of text lines using the level set method, most text lines are segmented correctly. A few lines may be broken into several segments due to the large horizontal gaps between neighboring words. Since the number of broken segments is small and the line segments are often quite long, it is easy to group them in a post processing step as follows.

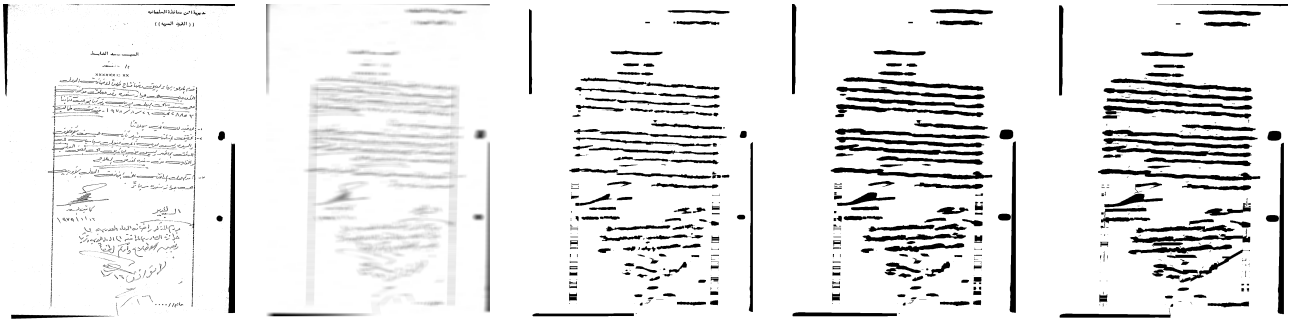
1. Calculate the orientation of each line segment using the minimal mean squared error.

2. Calculate the length of each line segment. If a text line is shorter than a threshold, its orientation is assumed to be horizontal.

3. Sort the text lines by length. From the longest text line, we try to extend the line to left or right using following rules: If there is another segment that was not already merged, their orientations are compatible and their horizontal gap is less than a threshold, we merge these two line segments . We apply this rule iteratively until no more line segments in left or right satisfying the merging condition.

4. Mark all segments of the resulting text line as "merged" so they will not be processed again.

By the end of this step, we obtain the major text line structures (Fig. 1e). Finally, we group the isolated connected components to their closest major text line if the



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

Figure 1. Illustration of the proposed text line segmentation method. (a) A handwritten example; (b) Blurred by a rectangle Gaussian window; (c) Initial estimate of text lines; (d) Result after 50 iterations of evolving using the level set method; (e) After connecting a few broken text lines in post-processing;

distance is not too large. Otherwise, we label those far from any major text line as noise. Fig. 2 shows some example of text line detection using different colors to distinguish neighboring text

3 Experiments

We tested our algorithm on more than 10,000 heterogeneous handwritten documents in different scripts, such as Arabic, Hindi, and Chinese (Fig. 2). In this section, we present a quantitative analysis of our algorithm. To visualize the detected structures, we use different colors to denote different text lines.

Our algorithm has three important parameters: 1) the estimate of text height, 2) the minimal number of characters to form a line, and 3) the iteration number of the level set evolving. The first one serves as the height of Gaussian window. It does not need to be very accurate as demonstrated in our following experiments. The second one is used to distinguish text lines and isolated characters, and we multiply first two parameters to obtain the width of Gaussian window. In our case we set it to three, which means one line consists of at least three characters.

3.1 Methodology and overall results

For a machine printed document, the text line segmentation results are often compared to the ground truth based on the bounding boxes. However, this simple approach does not apply for handwritten documents where text lines are curvilinear. Therefore, a polygon is used to represent the segmentation ground truth. We have ground-truthed 100 handwritten Arabic documents using the VIPER Ground Truth Editor [14].

The Hungarian algorithm [15] is used to find one-to-one correspondence between the detected and ground-truthed text lines by maximizing the number of shared pixels. We evaluate the performance based on the hit rate, which is defined as the number of shared pixels normalized by the total number of pixels of the ground-truthed lines. By using the pixel-level hit rate and 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 compare our approach with an improved connected component method in Doclib [4], and the results are shown in Table 1. Comparing to the hit rate of 66% of the connected component based method, we achieve a much better result with a 92% hit rate.

We can also evaluate the performance at the text line level. 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. There are a total of 2,691 ground-truthed lines, and our approach can detect 2,303 (85.6%) of them correctly, as shown in the last column of Table 1. At the text line level, the connected component based method performs significantly worse. Only 951 (35.3%) text lines are detected correctly.

Table 1. Quantitative comparison of our approach and a connected component based approach on 100 handwritten Arabic documents.

	Hit Rate	STD of Hit Rate	Detected Text Lines
Connected Component Based Approach	66%	0.25	951 (35.3%)
Our Method	92%	0.04	2,303 (85.6%)

We also analyze the failure examples of our algorithm. Most of the failure examples (e.g., Fig. 2e and 2f) are because two neighboring text lines overlap significantly. Other reasons include signatures, the correction in the gap between two lines, the severe noise introduced in the scanning step. Since two lines are connected in only few touched areas, a post-processing to segment them horizontally is necessary to improve the performance.

For simplicity, we use hit rate for text lines as the metric in the following discussion.

3.2 Robustness Test

3.2.1 Results for machine printed document

Fig. 4 shows the segmentation results on two machine printed documents without tuning parameters. Results show that our algorithm has a performance comparable

to traditional methods on similar machine printed documents.

3.2.2 Result for scripts

We have already showed some images in Fig. 2. We can calculate the hit rate to compare performance of different scripts. Fig. 3a shows that our algorithm has similar performance in 5 scripts. These results also show that our algorithm is script independent.

3.2.3 Result for skews

To illustrate how the skew degrades the performance, we can rotate an image using different angles and compare the accuracy. Fig. 5 shows results at three different rotation angles. The quantitative evaluation is shown in Fig. 3b. We show this document because of the orientation of text lines varies, and the gap between two lines are also different in many portions. The second and third figure show that the performance of our algorithm does not degrade much when orientation angle changes within [-10°, 10°]. For most deskewed handwritten documents, the skew falls into this range, so we claim that our algorithm can fit in most cases with a deskew module as preprocessing.

3.2.4 Results for character sizes

In most cases, we cannot determine the exact character size. Therefore, we test performance for different character size to evaluate the sensitivity. By using the same parameter, we can simulate this factor by resizing images. Fig. 3c shows accuracy for different scales. It shows our algorithm produces similar results under different character sizes.

3.2.5 Results for noise

By randomly flipping the pixels in binary images we can do some simulations with different probability. Fig. 6 shows three images with different noise level. Fig. 3d also gives a statistical analysis of accuracy in terms of different noise probability.

4 Conclusion

We proposed a new approach for text line detection by adopting a state-of-the-art image segmentation technique. We first convert a binary image to gray scale using a Gaussian window, which enhances text line structures. Text lines are extracted by evolving an initial estimate using the level set method.

Preliminary experiments show that our method is more robust compared to a bottom-up connected component based approach [4]. A comprehensive evaluation is performed and examples are shown. Examples show that our method is script independent. This has been qualitatively confirmed by testing it on handwritten documents in different languages, such as Arabic, English, Chinese, Hindi, and Korean. On machine printed documents, our approach has similar performance as traditional methods and is capable of processing a complex layout, therefore, it combines the advantages of the bottom-up and top-down approaches. Statistical results show that our algorithm produces consistent results under reasonable variation of skew angles, character sizes, and noise.

References

[1] 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*, pages 159–163, 2001.

[2] R. Manmatha and J. Rothfeder. A scale space approach for automatically segmenting words from historical handwritten documents. *IEEE Trans. Pattern Anal. Machine Intell.*, 27(8):1212–1225, 2005.

[3] A. Simon, J.-C. Pret, and A. Johnson. A fast algorithm for bottom-up document layout analysis. *IEEE Trans. Pattern Anal. Machine Intell.*, 19(3):273–277, 1997.

[4] S. Jaeger, G. Zhu, D. Doermann, K. Chen, and S. Sampat. DOCLIB: A software library for document processing. In *Proc. of SPIE vol. 6067*, pages 63–71, 2006.

[5] B. Yu and A. Jain. A robust and fast skew detection algorithm for generic documents. *Pattern Recognition*, 29(10):1599–1629, 1996.

[6] G. Nagy, S. Seth, and M. Viswanathan. A prototype document image analysis system for technical journals. *Computer*, 25(7):10–12, 1992.

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

[8] U. Pal and P.P. Roy. Multioriented and curved text lines extraction from Indian documents. *IEEE Trans. System, Man and Cybernetics – Part B: Cybernetics*, 34(4):1676–1684, 2004.

[9] A. Zahour, B. Taconet, P. Mercy, and S. Ramdane. Arabic hand-written text-line extraction. In *Proc. Int'l Conf. Document Analysis and Recognition*, pages 281–285, 2001.

[10] J. Sauvola, D. Doermann, and M. Pietikainen. Locally adaptive document skew detection. In *SPIE vol.3027*, pages 96–108, 1997.

[11] S. Osher and R. Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*. Springer, 2000.

[12] A. Rosenfeld and J. Pfaltz. Sequential operations in digital picture processing. *Journal of the Association for Computing Machinery*, 13(4):471–494, 1966.

[13] B. Sumengen. *Variational Image Segmentation and Curve Evolution on Natural Images*. PhD thesis, University of California, Santa Barbara, Sept. 2004.

[14] D. Doermann and D. Mihalcik. Tools and techniques for video performances evaluation. In *Proc. Int'l Conf. Pattern Recognition*, pages 167–170, 2000.

[15] G. Liu and R.M. Haralick. Optimal matching problem in detection and recognition performance evaluation. *Pattern Recognition*, 35(3):2125–2135, 2002.

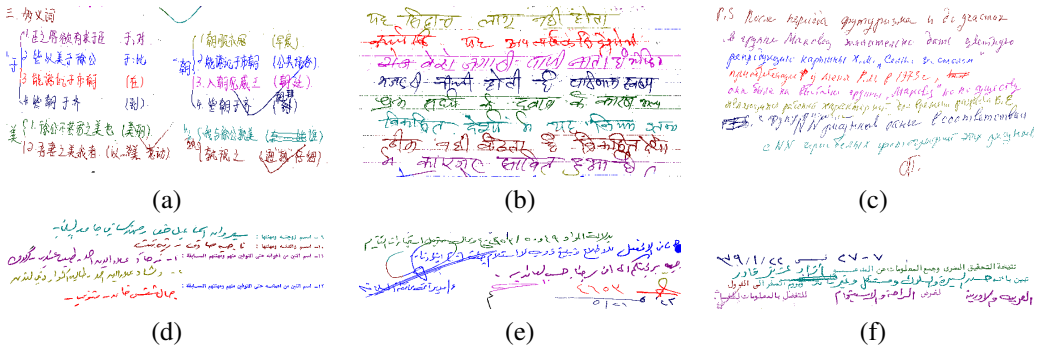


Figure 2. Color examples of text line segmentation. (a) Handwriting with an irregular document layout; (b) Handwritten text lines with background rule lines; (c) Curvilinear handwritten text lines; (d) A document mixing with handwritten and machine printed text lines; (e) and (f) Failure examples.

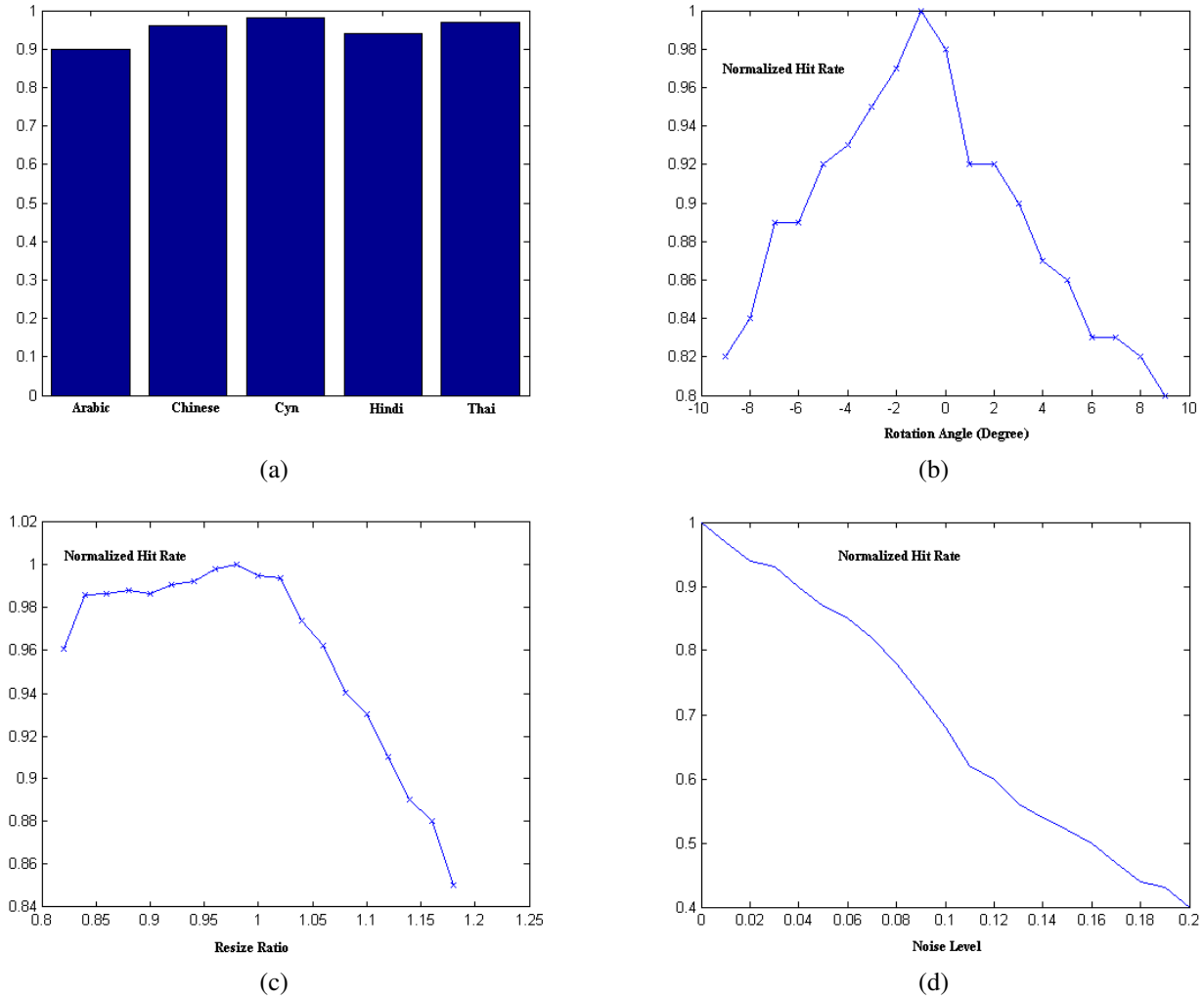


Figure 3. Robustness test. (a) Hit rate of different scripts. (b) Hit rate of different orientations. (c) Hit rate of different character sizes. (d) Hit rate of different noise levels. (Hit rate is normalized for (b), (c), and (d))

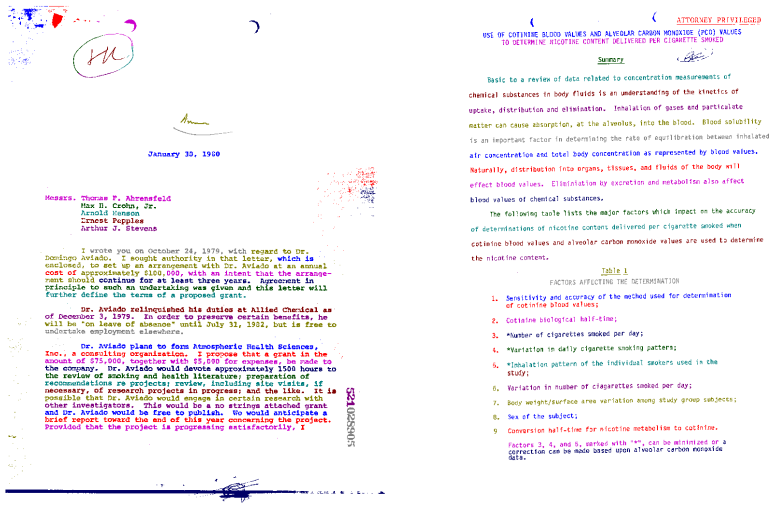
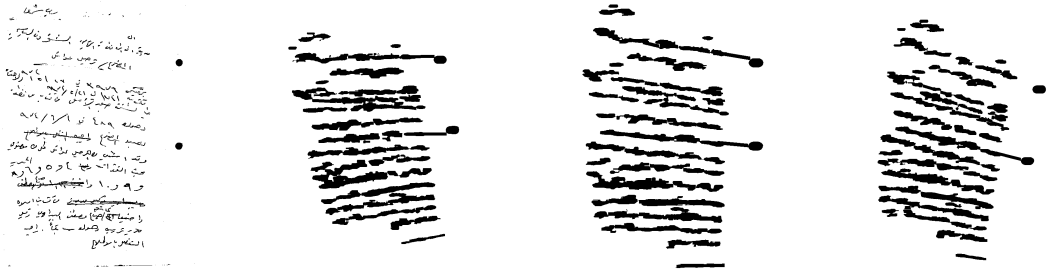
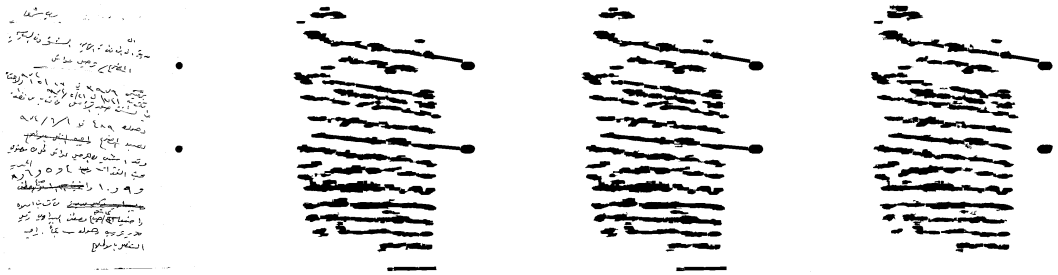


Figure 4. Segmentation results on two machine printed documents (labeled using different colors).



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

Figure 5. Segmentation results for documents with different orientations. (a) Original image, (b) Rotate - 10 °, (b) Rotate 0 °,(d) Rotate 10 °.



(a) (b) (c)

Figure 6. Segmentation results for documents with different noise levels. (a) Original image, (b) Probability = 0.02, (c) Probability = 0.07, (d) Probability = 0.15.