A Model-based Line Detection Algorithm in Documents

Yefeng Zheng, Huiping Li, David Doermann
Laboratory for Language and Media Processing
Institute for Advanced Computer Studies
University of Maryland, College Park, MD 20742-3275
E-mail: {zhengyf, huiping, doermann}@umiacs.umd.edu

Abstract

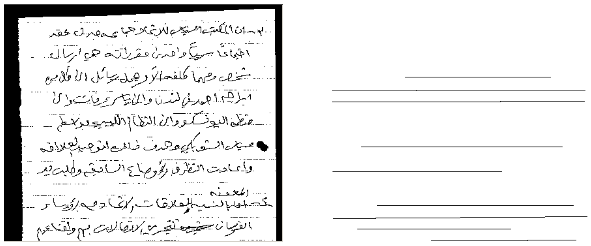
In this paper we present a novel model based approach to detect severely broken parallel lines in noisy textual documents. It is important to detect and remove these lines so the text can be segmented and recognized. We use Directional Single-Connected Chain, a vectorization based algorithm, to extract the line segments. We then instantiate a parallel line model with three parameters: the skew angle, the vertical line gap, and the vertical translation. A coarse-to-fine approach is used to improve the estimation accuracy. From the model we can incorporate the high level contextual information to enhance detection results even when lines are severely broken. Our experimental results show our method can detect 94% of the lines in our database with 168 noisy Arabic document images.

1 Introduction

It is not uncommon for background lines to exist in documents, touching or mixing with text. Figure 1(a) shows an Arabic document with background parallel lines and handwriting. These lines are printed on the paper as a guide for writers. After digitization however they can cause problems for segmentation and recognition algorithms. It is important that these lines be detected and removed before we feed the text into the Optical Character Recognition (OCR) engine.

1.1 Related Work

Line detection and removal are widely applied in table detection and interpretation [1, 2], engineering graph interpretation [3], and bank check/invoice processing [4, 5]. The line detection algorithms presented can be broadly classified as: Hough transform or vectorization based [6]. The Hough transform is a global approach with the ability to detect dashed and mildly broken lines, but is extremely time consuming [7]. To reduce the computational cost, a projection based method is proposed in [8] to search lines only near 0^o or 90^o. The algorithm is much faster than Hough transform, but it can only detect roughly horizontal or vertical lines. Vectorization based algorithms first extract vectors from the image, then merge vectors into lines, including methods such as BAG [1] and SPV [6]. Recently Zheng presented a novel vectorization based algorithm called the Directional Single-Connected Chain (DSCC) [2]. Each extracted DSCC represents a line segment and multiple non-overlapped DSCCs are merged into a line based on rules.



(a) (b)

Figure 1. Detection results of severely broken background lines. (a) A handwritten Arabic document image; (b)the line detection results.

These line detection algorithms work well on relatively clean documents with solid or mildly broken lines. In our task there are two challenges: 1) the lines are severely broken due to the low image quality, and 2) the lines are mixed with text, making the separation difficult. Figure 1(b) shows the line detection results using a vectorization (DSCC) based algorithm. We can see some lines are missed, and some are broken into several segments. It is very difficult, if not impossible, to detect those lines without contextual information. In forms analysis, most form cells are rectan-

The support of this research by the Department of Defense under contract MDA-9040-2C-0406 is gratefully acknowledged.

Proceedings of the Seventh International Conference on Document Analysis and Recognition (ICDAR 2003)
0-7695-1960-1/03 $17.00 © 2003 **IEEE**

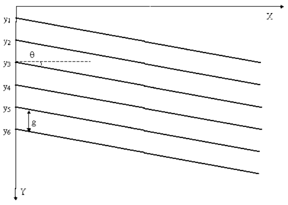


Figure 2. A model for a group of parallel lines

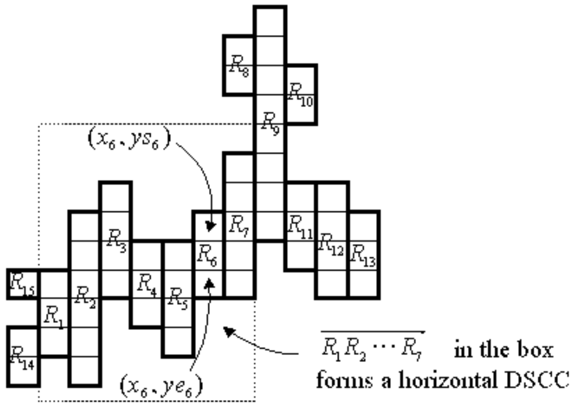


Figure 3. The definition of the horizontal DSCC

gular and composed of horizontal and vertical lines. Some work uses this a priori knowledge to correct the low level line detection errors [2, 9]. In both approaches, the contextual information is incorporated into hand tuned rules in an ad hoc way. In this paper, we use a model to incorporate the contextual information systematically to assist the broken line extraction.

1.2 Modeling

Horizontal parallel lines are often printed on paper so the users can write neatly. After digitization, these lines may be skewed and broken, but in our application we observe that 1) they are often parallel and 2) the gaps between any two neighboring lines are roughly equal. We therefore present a model for a group of parallel lines, $\mathcal{M}(\theta, g, y_1)$, where θ is the skew angle, g is the vertical line gap between two neighboring lines, and y_1 is the vertical translation (position of the left end point of the first line), as shown in Figure 2. The model considers the physical formation of these parallel lines, and can be used as context to extract them even when they are broken in noisy handwritten documents.

We assume all lines start from the left border and end at the right border. The actual line position can be identified by checking the black pixels. Since the x coordinate of the end point is always 0 or $w-1$ (w is the width of the image), in the following presentation, we use only the y coordinate (L_i, R_i) of the left and right end points respectively to represent line i. The vertical line gap between two neighboring lines i and j is defined as:

$$\begin{aligned} g_l &= |L_i - L_j| \\ g_r &= |R_i - R_j| \\ g &= (g_l + g_r)/2 \end{aligned} \quad (1)$$

For ideal parallel lines, $g_l = g_r = g$. The position of any line i, $i = 0, 1, 2...$, is calculated as:

$$\begin{aligned} L_i &= y_1 + i \times g \\ R_i &= L_i + w \times \text{ctan}(\theta) \end{aligned} \quad (2)$$

The remainder of this paper is organized as follows. In the next section we describe the details of our algorithm, including pre-processing, model parameter estimation and post processing. We present our experimental results in Section 3 and conclude our work with some discussion and future work.

2 Approach

2.1 Extraction of DSCC

We start from the extraction of DSCCs presented in [2]. Each DSCC is viewed as a line segment. Horizontal and vertical DSCCs are extracted to describe horizontal and vertical line segments respectively. A horizontal DSCC C_h consists of a black pixel run-length array $\overline{R_1 R_2 \cdots R_m}$, where each R_i is a vertical run-length with one pixel width:

$$R_i(x_i, ys_i, ye_i) = \{(x,y) | \forall \mathrm{I}(x,y) = 1, \text{ for } x = x_i, \quad y \in [ys_i, ye_i], \text{and } \mathrm{I}(x_i, ys_i - 1) = \mathrm{I}(x_i, ye_i + 1) = 0\} \quad (3)$$

$\mathrm{I}(x,y)$ is the value of pixel (x,y) with 1 representing black pixels and 0 representing white pixels; x_i, ys_i, and ye_i are the x, starting, and ending y coordinates of R_i respectively. If two neighboring run-lengths R_i and R_{i+1} are singly connected, we merge them as a part of a DSCC. Single connection means that on each side of R_i, there is one and only one run-length connected with it. All the singly connected run-lengths are merged to form a DSCC, as shown in Figure 3. The definition of a vertical DSCC, C_v, is similar. The detailed definition and extraction algorithm can be found in [2]. In our work, we are primarily interested in horizontal lines, hence only horizontal DSCCs are extracted.

2.2 Pre-processing

The extracted DSCCs include line segments, noise and handwritten strokes. Before we merge DSCCs into lines, we filter those DSCCs generated by noise and handwriting strokes. We observed that a horizontal line segment often has a small skew angle, and large aspect ratio. Therefore, we use an ellipse to model the shape of a DSCC, and calculate the skew angle θ, the first and second axes a and b of each DSCC as following:

$$\mu_{pq} = \sum_x \sum_y (x-\bar{x})^p (y-\bar{y})^q \mathrm{I}(x,y) \tag{4}$$

$$\theta = 0.5\tan^{-1}\left(\frac{2\mu_{11}}{\mu_{20}-\mu_{02}}\right) \tag{5}$$

$$a = \sqrt{\frac{2[\mu_{20}+\mu_{02}+\sqrt{(\mu_{20}-\mu_{02})^2+4\mu_{11}^2}]}{\mu_{00}}} \tag{6}$$

$$b = \sqrt{\frac{2[\mu_{20}+\mu_{02}-\sqrt{(\mu_{20}-\mu_{02})^2+4\mu_{11}^2}]}{\mu_{00}}} \tag{7}$$

Where $\bar{x}$ and $\bar{y}$ are the means of x and y coordinates and u_{pq} is the central moments. We only preserve those DSCCs satisfying:

1) Skew angle $\theta \in [-45^o, 45^o]$.

2) Aspect ratio $a/b > T$, where T is a threshold determined experimentally.

Figure 4(a) shows an original image and Figure 4(b) shows the corresponding filtered result. We can see most text strokes are filtered and the background line segments are well preserved.

2.3 The Estimation of Model Parameters

After filtering, we merge neighboring DSCCs into lines, as shown in Figure 4(c). We can see the lines are low quality and broken, and need to use the model to refine the detection results.

2.3.1 Skew Angle Estimation

A two-step coarse to fine method is used to estimate the skew angle. In the coarse estimation, we construct a weighted angle histogram of all extracted lines (shown in Figure 4(c)) as follows:

```
Clear all entries of the histogram h
For each line i do
    θ_i = skew angle of line i
    Weight w_i = length of line i
    h(θ_i) = h(θ_i) + w_i
End
```

The length of the line is used as the weight so the long lines can be emphasized. The angle, θ_c, with the largest count in the histogram is taken as the coarse estimate of the skew angle. We refine the estimation further by performing projection along the angles at a small range around θ_c, as shown

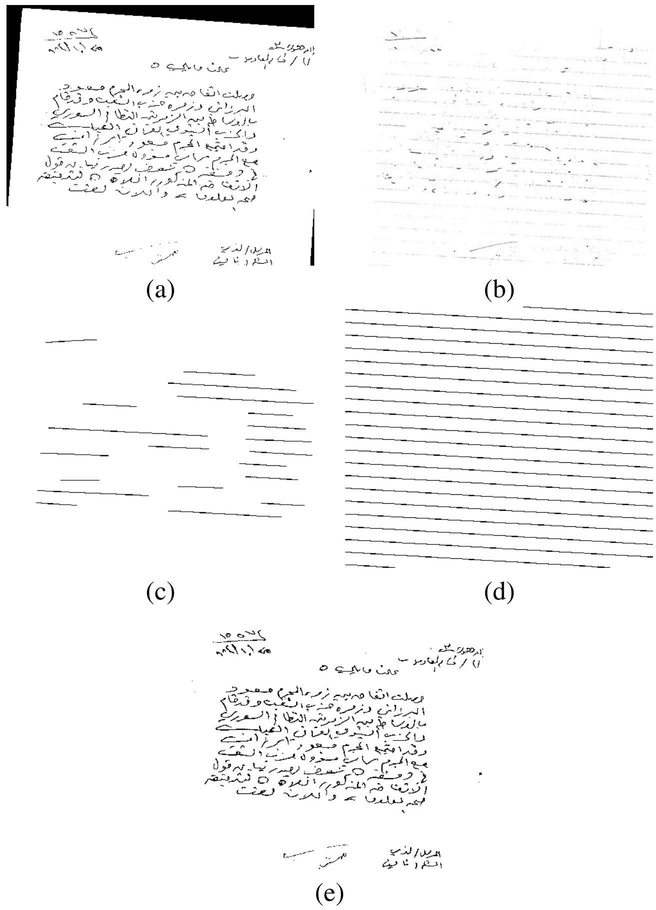
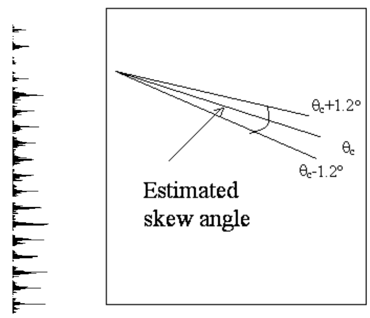


Figure 4. An example of background parallel line detection. (a) Original document image; (b) after filtering; (c) lines detected after merging neighboring DSCCs; (d) line detection results using the model; (e) after line removal.



(a) (b)

Figure 5. Refinement of the skew angle estimation. (a) Horizontal projection along the skew angle; (b) searching range.

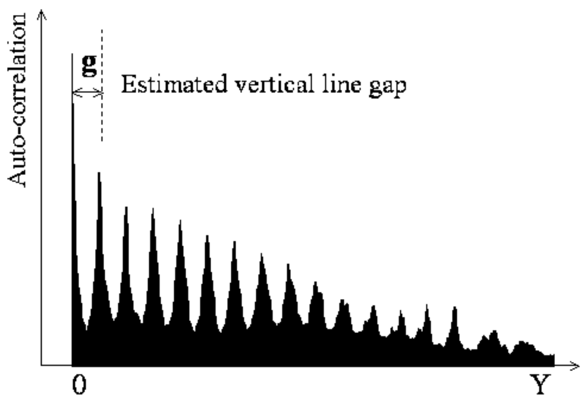


Figure 6. Vertical line gap estimation based on the auto-correlation of the projection.

in Figure 5. The angle, $\hat{\theta}$, which maximizes the variance of the projection is the refined estimate:

$$\hat{\theta} = \arg \max_{\theta \in [\theta_c - 1.2^o, \theta_c + 1.2^o]} Var(h(y, \theta)) \quad (8)$$

where $h(y, \theta)$ is the projection along the skew angle θ. Experiments conducted on our database containing 168 Arabic documents show the errors of coarse skew estimate are within the range of $[-1.16^o, 1.17^o]$, and reduced to $[-0.56^o, 0.18^o]$ after refinement.

2.3.2 Vertical Line Gap Estimation

Under the assumption of relatively consistent vertical line gaps between neighboring lines, the projection of background parallel lines is a periodic signal and the period is the vertical line gap. We use an auto-correlation based approach to estimate the period of the projection. The auto-correlation of a signal x, with n samples $x(0), x(1), \cdots, x(n-1)$, is defined as:

$$R(l) = \sum_{i=0}^{n-1-l} x(i)x(i+l) \quad (9)$$

The distance between the first two peaks of the auto-correlation is taken as the vertical line gap, as shown in Figure 6.

To see how accurate our estimate is, we compare the estimate with the actual vertical line gap of the groundtruthed lines. The page level estimation error is defined as the average of estimation error of all vertical line gaps on a document page. For 168 images, most page level estimation errors are within 0.5 pixels. The maximum page level estimation error is 1.3 pixels due to the large vertical line gap variance.

2.3.3 Vertical Translation Estimation

We can use the position of the largest peak of the horizontal projection as the estimate of the vertical translation y_1, but it is not robust when all background lines are severely broken. Instead we search a position within $(0, g)$ which maximizes the periodic summation of the projection as y_1:

$$y_1 = \arg \max_{0<y<g} \sum_i h(i \times g + y, \hat{\theta}) \quad (10)$$

2.4 Post-Processing

Our model can detect most of the background parallel lines, but problems may appear due to the distortion introduced by printing, photocopying, and/or improper positioning of the document during scanning. This distortion may affect the estimate of the skew angle and the vertical line gap. As a post-processing, we vary the position of the left and right end points of line (L, R) inside a small range to search for a new position $(\hat{L}, \hat{R})$ which maximizes the number of black pixels on the line.

$$(\hat{L}, \hat{R}) = \arg \max_{\substack{L-E<l<L+E \\ R-E<r<R+E}} \#\text{ of black pixels on line (l, r)} \quad (11)$$

Where (L, R) and $(\hat{L}, \hat{R})$ are the line positions before and after refinement, and E is the search range with the size of 10 pixels in our experiments.

3 Experiments

We obtained 168 Arabic document images with a total of 3,922 groundtruthed lines, most of which are severely broken. Line detection accuracy can be evaluated at the pixel level and the line level [10]. The pixel level evaluation compares the difference of the pixels between groundtruthed and detected lines. It is straightforward and objective, but the groundtruth at the pixel level is extremely expensive when lines are broken, noisy, and distorted. Therefore, we evaluate the algorithm at the line level. The first step is to define the evaluation metrics. The distance between a detected line and a groundtruth line is used to measure how well they match. For evaluation, we extend the end points of all lines (both detected and groundtruthed) to the left and right image borders. Assume a detected line is (L_d, R_d) and the corresponding groundtruthed line is (L_g, R_g), then the distance D is defined as the maximal mismatch between two end points:

$$D = \max(|L_d - L_g|, |R_d - R_g|) \quad (12)$$

A detected line matches a groundtruthed line if the distance $D < g/3$, where g is the vertical line gap. If there is more

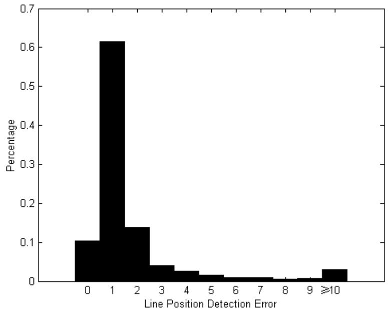


Figure 7. Histogram of background line detection errors.

than one detected line matching a groundtruthed line, or vice verse, then only the match with the minimal distance is kept. The detection error of a groundtruthed line is defined as the distance to its matched detected line, as in Equation (12). The histogram of the detection errors is shown in Figure 7. Over 84% of the groundtruthed lines are detected with errors less than 3 pixels, 94% are detected with the accuracy of less than 5 pixels, and we did not miss any groundtruthed lines. A false alarm happens if a detected line can not match any groundtruthed lines. Most of the false alarms are generated because our model detected severely broken lines which are not groundtruthed based on the subjective judgment of the groundtruther. Since our ultimate goal is to remove background lines to achieve a clean document, these false alarms do not create problems for line removal, as no useful information is removed in this case. Figure 4(d) shows an example of background line detection results with the model. Compared with Figure 4(c), we can see our model based approach gets much better results.

4 Conclusion and Future Work

In this paper we have presented a novel approach to detect severely broken parallel lines in noisy documents. Our method is based on a model to incorporate high level constraints into a general line detection algorithm. Experiments show our method can detect 94% of lines in the database we collected.

After line detection, we can remove these detected lines to achieve a cleaned version of the document. Figure 4(e) is the result of Figure 4(a) after we remove the black pixels on the line and filter the noise. While the result is encouraging, we find some text strokes touching the detected lines are removed erroneously. We are investigating a more robust algorithm to remove the lines and reserve the text strokes. We will report the result in future research.

References

[1] B. Yu and A. K. Jain. A generic system for form dropout. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 18(11):1127–1131, 1996.

[2] Y. Zheng, C. Liu, and X. Ding. Form frame line detection with directional single-connected chain. In *Proc. Int'l Conf. Document Analysis and Recognition*, pages 699–703, 2001.

[3] D. Dori, Y. Liang, and J. Dowell. Sparse-pixel recognition of primitives in engineering drawings. *Machine Vision and Application*, 6:69–82, 1993.

[4] F. Cesarini, M. Gori, and S. Marinai. INFORMys: A flexible invoice-like form-reader system. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 20(7):730–745, 1998.

[5] Y. Y. Tang, C. Y. Suen, and C. D. Yan. Financial document processing based on staff line and description language. *IEEE Trans. Systems, Man and Cybernetics*, 25(5):738–753, 1995.

[6] W. Liu and D. Dori. From raster to vectors: Extracting visual information from line drawings. *Pattern Analysis and Application*, 2(1):10–21, 1999.

[7] J. Illingworth and J. Kittler. A survey of the Hough transform. *CVGIP*, 44:87–116, 1988.

[8] J. Liu, X. Ding, and Y. Wu. Description and recognition of form and automated form data entry. In *Proc. Int'l Conf. Document Analysis and Recognition*, pages 579–582, 1995.

[9] O. Hori and D. Doermann. Robust table-form structure analysis based on box-driven reasoning. In *Proc. Int'l Conf. Document Analysis and Recognition*, pages 218–221, 1995.

[10] W. Liu and D. Dori. A protocol for performance evaluation of line detection algorithms. *Machine Vision and Application*, 9(5/6):57–68, 1997.