A Model-based Ruling Line Detection Algorithm for Noisy Handwritten Documents

Jin Chen
Dept. of Computer Science & Engineering
Lehigh University
Bethlehem, PA 18015, USA
jic207@cse.lehigh.edu

Daniel Lopresti
Dept. of Computer Science & Engineering
Lehigh University
Bethlehem, PA 18015, USA
lopresti@cse.lehigh.edu

***Abstract*—Ruling lines are commonly used to help people write neatly on paper. In document image analysis, however, they create challenges for handwriting recognition and writer identification. In this paper, we model ruling line detection as a multi-line linear regression problem and then derive a globally optimal solution giving the Least Square Error. We demonstrate the efficacy of the technique on both synthetic and real datasets. A comparative study shows that our algorithm outperforms a previously published method on the public Germana dataset.**

***Keywords*-handwritten documents; ruling line detection;**

I. Introduction

The need for line recognition arises in various document analysis applications, e.g., form/invoice processing [1], engineering drawing conversion [2], musical score capture [3], and layout analysis [4]. Many techniques work well on clean images of good quality, but their performance deteriorates when lines are badly broken due to light printing or scanning, low input resolutions, or significant overlap with handwriting on the page [5].

Zheng, *et al.*, presented a stochastic model-based parallel line detection algorithm that incorporates context [4]. Rather than treat peaks on the projection profile as line positions, they model lines using a Hidden Markov Model so that inter-line constraints can be incorporated.

In this work, we propose a model-based ruling line detection algorithm that takes advantages of page-level attributes, *e.g.*, consistent spacing, skew angle, thickness, and length. First, we introduce the framework of multi-line linear regression and derive a globally optimal solution giving the Least Square Error (LSE). Then we describe a Hough transform variant for extracting line segments and the "Basic Sequential Algorithmic Scheme (BSAS)" for grouping line segments. After initial line clusters have been determined, potential missed lines are identified by exploiting the model-based nature of the method. We conduct a performance analysis on the basis of individual model attributes, rather than attempting to define a single metric that combines all attributes together as in [6].

The remainder of this paper is organized as follows: in Section II, we introduce the multi-line linear regression model. Then we describe our ruling line detection algorithm in Section III, and our experimental setup in Section IV. We present experimental results in Section V, and finally conclude in Section VI.

II. Multi-line Linear Regression

The multi-line linear regression model assumes:

i) Ruling lines are straight.
ii) Lines exhibit consistent spacing and skew.
iii) The association between points and lines is available.
iv) The number of lines k is known.
v) No lines are missing between the first and last lines.

It is important to clarify that properties iii), iv), and v) are guaranteed by the first phase of our algorithm, and are not fundamental limitations on our formulation of the problem.

Since the number of lines k and the point-to-line association are provided, the points set $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ can be further formulated as $\{(x_{i,j}, y_{i,j}) \mid i = 0, \ldots, k-1; j = 1, \ldots, n_i; \sum_{i=0}^{k-1} n_i = n\}$. In addition, the ruling line are parallel and have the same spacing, so we rewrite the normal form of the line equation as:

$$\beta_0 + i\beta_1 + \beta_2 x_{i,j} + \epsilon_i = y_{i,j} \tag{1}$$

where $i = 0, \ldots, k-1$, $j = 1, \ldots, n_i$, and $\sum_{i=0}^{k-1} n_i = n$. ϵ_i is a random error component. β_0 is the y-intercept of the 0-th line, β_1 is the spacing between lines, and β_2 is the skew angle.

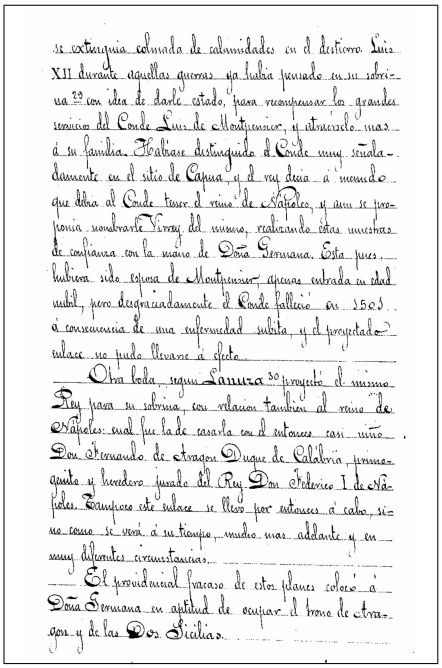
Next, we rewrite the *residual function* as:

$$\begin{aligned} f \overset{\text{def}}{=} \epsilon^2 &\overset{\text{def}}{=} \|\boldsymbol{b} - A\,\hat{\mathbf{x}}\|^2 \\ &= \sum_{i=0}^{k-1}\sum_{j=1}^{n_i}(y_{i,j} - \hat{\beta}_0 - i\hat{\beta}_1 - \hat{\beta}_2 x_{i,j})^2 \end{aligned} \tag{2}$$

To minimize the residual, we take the partial derivatives of f with respect to $\hat{\beta}_0, \hat{\beta}_1, \hat{\beta}_2$, respectively (denoting $\sum_{i=0}^{k-1}\sum_{j=1}^{n_i}$ as $\boldsymbol{\Sigma}$).

$$\begin{cases} \frac{\delta f}{\delta \hat{\beta}_0} = -\boldsymbol{\Sigma} 2(y_{i,j} - \hat{\beta}_0 - i\hat{\beta}_1 - \hat{\beta}_2 x_{i,j}) = 0 \\ \frac{\delta f}{\delta \hat{\beta}_1} = -\boldsymbol{\Sigma} 2k(y_{i,j} - \hat{\beta}_0 - i\hat{\beta}_1 - \hat{\beta}_2 x_{i,j}) = 0 \\ \frac{\delta f}{\delta \hat{\beta}_2} = -\boldsymbol{\Sigma} 2x_i(y_{i,j} - \hat{\beta}_0 - i\hat{\beta}_1 - \hat{\beta}_2 x_{i,j}) = 0 \end{cases} \tag{3}$$

1520-5363/11 $26.00 © 2011 IEEE
DOI 10.1109/ICDAR.2011.89



(a)

Figure 1: A sample from the Germana dataset.

Solving these equations, we have:

$$\begin{bmatrix} \boldsymbol{\Sigma}1 & \boldsymbol{\Sigma}i & \boldsymbol{\Sigma}x_{i,j} \\ \boldsymbol{\Sigma}i & \boldsymbol{\Sigma}i^2 & \boldsymbol{\Sigma}i\ x_{i,j} \\ \boldsymbol{\Sigma}x_{i,j} & \boldsymbol{\Sigma}i\ x_{i,j} & \boldsymbol{\Sigma}x_{i,j}^2 \end{bmatrix} \times \begin{bmatrix} \hat{\beta}_0 \\ \hat{\beta}_1 \\ \hat{\beta}_2 \end{bmatrix} = \begin{bmatrix} \boldsymbol{\Sigma}y_{i,j} \\ \boldsymbol{\Sigma}i\ y_{i,j} \\ \boldsymbol{\Sigma}x_{i,j}\ y_{i,j} \end{bmatrix} \quad (4)$$

The 3-by-3 *design* matrix is a symmetric matrix with positive entries. From physical considerations where the number of lines and the point-to-line association are available, we can expect a unique solution $\hat{\mathbf{x}} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b}$ to $\mathbf{Ax} = \mathbf{b}$.

III. Model-based Ruling Line Detection

Our model-based ruling line detection algorithm employs the following assumptions: (1) Pages exhibit salient, although not necessarily continuous, ruling line segments (Figure 1), and (2) Ruling lines are parallel and have consistent spacing, thickness, and length.

A. A Variant of The Hough Transform

The classical Hough Transform projects each point onto a set of sinusoidal curve points in the (ρ, θ) plane (the Hough Space):

$$\rho = x_i\ cos\theta + y_i\ sin\theta \quad (5)$$

where $\theta \in [0, 2\pi)$. We adopt an effective variant in our work [7]. First, in each iteration we select a point randomly from the remaining point set, and then compute its sinusoidal curve in the Hough space and update the accumulation matrix. If the guard of the current maximum votes is larger than the threshold, then we search in each direction from the current position for the end points of the line segment. Since ruling lines may be degraded in the page image, short gaps (up to five pixels) are tolerated during the search. Once the search stops, we record the coordinates of the end points for the line segment, remove these points from the accumulation matrix, and proceed.

B. Sequential Clustering

After the Hough transform, we possess a set of line segments specified by their end points. Denote $\mathcal{T}$ as the threshold of dissimilarity between clusters, and $\mathcal{Q}$ as the maximum number of clusters. In our experiments, the dissimilarity measure is the ρ-value distance between line segments and we set $\mathcal{T} = 10$ and $\mathcal{Q} = 32$ empirically.

We outline the "Basic Sequential Algorithmic Scheme" (BSAS) in Algorithm 1.

Algorithm 1: Basic Sequential Algorithmic Scheme [8]

Data: $G = \{\mathbf{x_i} : i = 1, \ldots, N\}$: a set of line segments.
Result: m: number of clusters acquired.
begin
  $m = 1$
  $C_m = \{x_i\}$
  for $i = 2$ **to** N **do**
    Find $C_j : \boldsymbol{d}(x_i, C_j) = min_{1 \le p \le m} \boldsymbol{d}(x_i, C_p)$
    if $\boldsymbol{d}(x_i, C_j) > \mathcal{T}$ **and** $(m < \mathcal{Q})$ **then**
      $m = m + 1$
      $C_m = \{x_i\}$
    else $C_j = C_j \cup \{x_i\}$

After BSAS clustering, we estimate the ruling line spacing by building a histogram containing votes for spacing values between two consecutive clusters. Since ruling lines in the Germana dataset are usually broken, we select the minimum spacing between clusters. This value is temporary – in a later stage we update it by re-computing the spacing globally and, hence, more precisely.

C. Single Line Fitting

After combining close clusters, we now have a good estimation of which line segments belong to which cluster. In this stage, we employ linear regression on each cluster. We compute a temporary skew angle by averaging all β_2 values. Again, this skew angle is tentative and will be fine-tuned by the multi-line linear regression in a later stage. The ruling line thickness $\mathcal{H}$ is computed by first building a histogram of vertical run-lengths along each line. We then examine the histogram and select the most frequent bucket as the thickness $\mathcal{H}$.

D. Reasoning About Missing Lines

For degraded documents, it is common for line segments to be missed by the Hough transform. To address this, we

traverse the clusters checking for whether two consecutive clusters have a larger than expected spacing. In such cases, we hypothesize that there may be missed ruling lines in between. We estimate the positions of missing lines by considering the spacing and the skew angle, then we scan along the missing lines to collect sample points for further regression. This procedure is depicted in Algorithm 2.

The subroutine *ScanZones* specifies "North" or "South" for the scanning direction, and employs a "strict" or "relaxed" criterion for collecting evidence. For the strict criterion, we decide that a ruling line exists only if we collect more than $\mathcal{T}_1$ sample points from the scanning area. For the relaxed criterion, we do not set such a threshold. We apply the strict criterion at the topmost and bottommost lines on the page, and the relaxed criterion for all other lines. The rationale is that if we want to add missing lines to the top/bottom of the current candidate list, we need strong evidence, otherwise we can derive their positions from the spacing between existing lines. In this way, we iteratively scan for missing lines until the point count is lower than the threshold or the process reaches the edge of the image boundary. In our experiments, $\mathcal{T}_1 = 0.2 \times page\_width$.

Algorithm 2: Find Missing Lines.

Data: A list of lines: $lines_{old}[m]$. The temporary spacing between lines: s.
Result: An updated list of lines: $lines_{new}[m^{'}]$.
begin
  $lines_{new}$ = ScanZones($lines_{old}[0]$, s, "North," "strict")
  for $i = 0$ **to** $m - 1$ **do**
    $space = lines_{old}[i+1].\rho - lines_{old}[i].\rho$
    $count = space/s$
    $local\_s = space/count$
    if $abs(space) > 1.5\ s$ **then**
      $lines_{new}$ = ScanZones($lines_{old}[i]$, $local\_s$, "South," "relaxed")
  $lines_{new}$ = ScanZones($lines_{old}[m]$, s, "South," "strict")

E. Computing Model Parameters

At this stage in the process, we have satisfied all of the prerequisites for Eq. 4 in Section II. Solving the equation, we obtain the estimated parameter vector $\hat{\boldsymbol{\beta}}$. Next, we update the linear equation for each cluster. Then for each cluster, we scan the areas that extend from the leftmost and rightmost points. Any newly discovered sample points are collected as the new starting and ending points for that line. We use the maximum line length as the ruling line length $\mathcal{L}$. At the same time, we can determine the starting position of the first ruling line $\mathcal{P}(x_p, y_p)$.

Algorithm 2 relies on the local spacing estimate to find missing lines, so it might not be fully reliable. Hence, we run another round of scanning using the global spacing estimate. A difference is that now $\mathcal{T}_2 = 0.01 \times page\_width$ for the degraded dataset Germana. If there are additional ruling lines detected at this stage, we update the corresponding model parameters: i.e., the number of ruling lines $\mathcal{K}$ and the starting position of the first line $\mathcal{P}(x_p, y_p)$.

To summarize, the model parameters determined by the algorithm are: the starting point of the first line $\mathcal{P}(x_p, y_p)$, the length $\mathcal{L}$, the thickness $\mathcal{H}$, the skew angle $\boldsymbol{\beta_2}$, the number of lines $\mathcal{K}$, and the spacing $\boldsymbol{\beta_1}$. We denote them as $\Theta = (\mathcal{P}(x_p, y_p), \mathcal{L}, \mathcal{H}, \boldsymbol{\beta_2}, \mathcal{K}, \boldsymbol{\beta_1})$.

IV. Experimental Setup

A. Data Preparation

We evaluated our ruling line algorithm on both synthetic and real datasets. First, as a simple test of correctness, we synthesized a dataset that contained only ruling lines using pre-determined parameter settings. One subset consisted of 11 pages, each containing 20 ruling lines possessing the same length, thickness, and spacing, but a different skew angle ranging from $[-1.0°, 1.0°]$ with a step size of $0.2°$. The other subset consisted of 10 pages where the lines had the same length, thickness, and skew angle, but the total number of lines ranged in the interval [10, 20), with the spacing varying as well. The ground-truth for this dataset was generated directly from the model parameters.

We also used one real dataset, the Germana collection [9], for performance evaluation. The ground-truth in this case was generated using the Badcat annotation tool developed at Lehigh. Employing a model-based graphical user interface, Badcat allows users to label ruling lines on a page quickly with a few simple "point-click-and-drag" operations. From the Germana dataset, we randomly selected 20 pages as the test set, and another 20 pages as the training set so that we could compare our method to Zheng, *et al.*'s Hidden Markov Model-based approach. A breakdown of datasets is listed in Table I.

Table I: A breakdown of datasets.

Dataset	pages	lines/page	page size
rotation-vary	11	20	$816w \times 1056h$
spacing-vary	10	$[10, 20)$	$816w \times 1056h$
Germana	20	24	$1420w \times 2120h$

B. Performance Evaluation Metric

The performance of line detection algorithms is generally measured in one of two ways: pixel-level metrics and object-level metrics [6]. Pixel-level metrics, including *precision*, *recall*, and *F-Score*, are intuitive measurements. However, pixel-accurate ground-truth is difficult to acquire because

labeling a page image at this level is tedious and subjective, especially when the lines are severely degraded. On the other hand, researchers have proposed a variety of object-level metrics. Unfortunately, compound metrics appear less effective in highlighting performance differences between algorithms (e.g., $Q_v(c)$ in Liu and Dori's work [6]).

Instead of proposing a compound metric that attempts to combine all parameters into a single value, we chose to measure directly the discrepancies between the computed parameters and the ground-truth where $\Theta = (\mathcal{P}(x_p, y_p), \mathcal{L}, \mathcal{H}, \boldsymbol{\beta_2}, \mathcal{K}$, and $\boldsymbol{\beta_1})$.

C. Post-processing Zheng, et al.'s Algorithm [4]

We treated Zheng, *et al.*'s algorithm as a black box which outputs a list of detected line segments. Some minor post-processing was necessary, however, to make the results comparable to ours.[1]. The number of lines $\mathcal{K}$ was taken intuitively to be the size of the list. The spacing $\boldsymbol{\beta_1}$ was detemined by the largest bucket in the spacing histogram. The skew angle $\boldsymbol{\beta_2}$ was computed as the average for all the line segments. The thickness $\mathcal{H}$ was acquired as explained in Section III-C. Finally, $\mathcal{P}(x_p, y_p)$ and $\mathcal{L}$ are computed exactly as described in Section III-E.

V. Experimental Results

We plot intermediate results from the model-base ruling line detection pipeline in Figure 2. Images from the Germana dataset are lower quality, so most ruling lines are broken and difficult even for humans. As we can see from Figure 2a, line segments tend to be incomplete and sparse. Running Algorithm 2, we recovered several missing lines, as shown in Figure 2b. At this stage, all lines shown in Figure 2c were used for the multi-line linear regression. Finally, using the skew angle and the line spacing, we successfully detected a missing line at the top of the page, as shown in Figure 2d.

To compare performance, we computed the error between an algorithm's output and the ground-truth:

$$D[i] = M_{algorithm}[i] - M_{ground-truth}[i] \qquad (6)$$

where $D[]$ is the error vector and $M_{(\cdot)}[]$ corresponds to the output or the ground-truth. We computed the mean and the standard deviation for each dataset. These results are organized in Table II. Note that there are two different synthetic subsets as explained earlier, but we summarize the performance here with a single entry in the table.

As might be expected, the synthetic dataset proved relatively easy: the mean and the standard deviation of the errors are much lower than those for the Germana dataset. We observed relatively large errors in terms of the σ-values for Germana. This is due largely to the degraded quality of the documents and the relatively low scanning resolution, causing many ruling lines to be thin and/or broken. In many cases it is difficult to identify the ruling lines precisely.

We also ran Zheng, *et al.*'s algorithm on Germana, randomly selecting 20 pages for HMM training. Table II shows the error statistics for the two approaches. As shown by the error means and standard deviations for the various parameters, our method performs better on Germana.

Figure 3 shows a sample result generated by Zheng, *et al.*'s algorithm. We also display our own results for comparison. For degraded images such as those in Germana, our algorithm manages to detect light and broken ruling lines. We consider this to be a compelling demonstration of the power of the model-based approach.

VI. Conclusion

In this paper, we introduced a model-based ruling line detection algorithm that requires no supervised learning. We first formulated the problem in the framework of multi-line regression and then derived a globally optimal solution giving the Least Square Error. Next, we introduced procedures for extracting line segments and detecting missing lines. Finally, we demonstrated the efficacy of our approach by comparing it to another method from the literature on two datasets, one synthetic and the other real page images.

Acknowledgment

We thank Dr. Yefeng Zheng for providing his parallel line detection package for comparison purposes. This work is supported by a DARPA IPTO grant administered by Raytheon BBN Technologies.

References

[1] J. Liu and A. Jain, "Image-based form document retrieval," *Pattern Recognition*, vol. 33, no. 3, pp. 503–513, 2000.

[2] D. Dori and W. Liu, "Sparse pixel vectorization: an algorithm and its performance evaluation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 3, pp. 202–215, 1999.

[3] D. Blostein and H. Baird, "A critical survey of music image analysis," in *Structured Document Image Analysis*, e. H. Baird, Ed., 1992, pp. 405–434.

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

[5] H. Cao and V. Govindaraju, "Handwritten carbon form preprocessing based on Markov random field," in *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

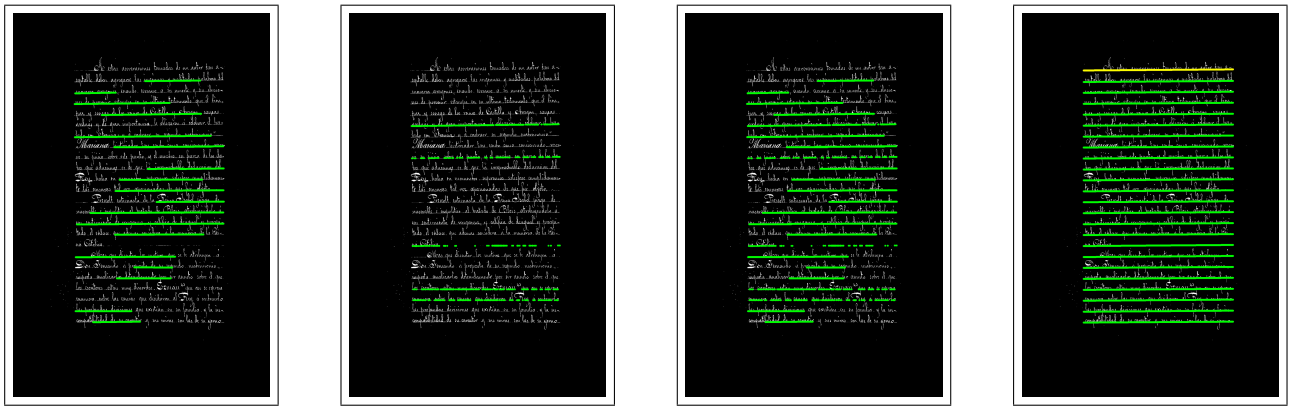
[6] W. Liu and D. Dori, "A protocol for performance evaluation of line detection algorithms," *Machine Vision And Applications*, vol. 9, pp. 240–250, 1997.

[7] J. Matas, C. Galambos, and J. Kittler, "Robust detection of lines using the progressive probabilistic hough transform," *Computer Vision and Image Understanding*, vol. 78, pp. 119–137, 2000.

[8] S. Theodoridis and K. Koutroumbas, *Pattern Recognition*. Academic Press, 2009.

[9] D. Pérez, L. Tarazón, N. Serrano, F. Castro, O. R. Terrades, and A. Juan, "The GERMANA database," in *Proc. of the International Conference on Document Analysis and Recognition*, 2009, pp. 301–305.

[1] We also observed some memory issues when executing Zheng, *et al.*'s code, so the execution was carefully supervised and the outputs were generated one at a time.

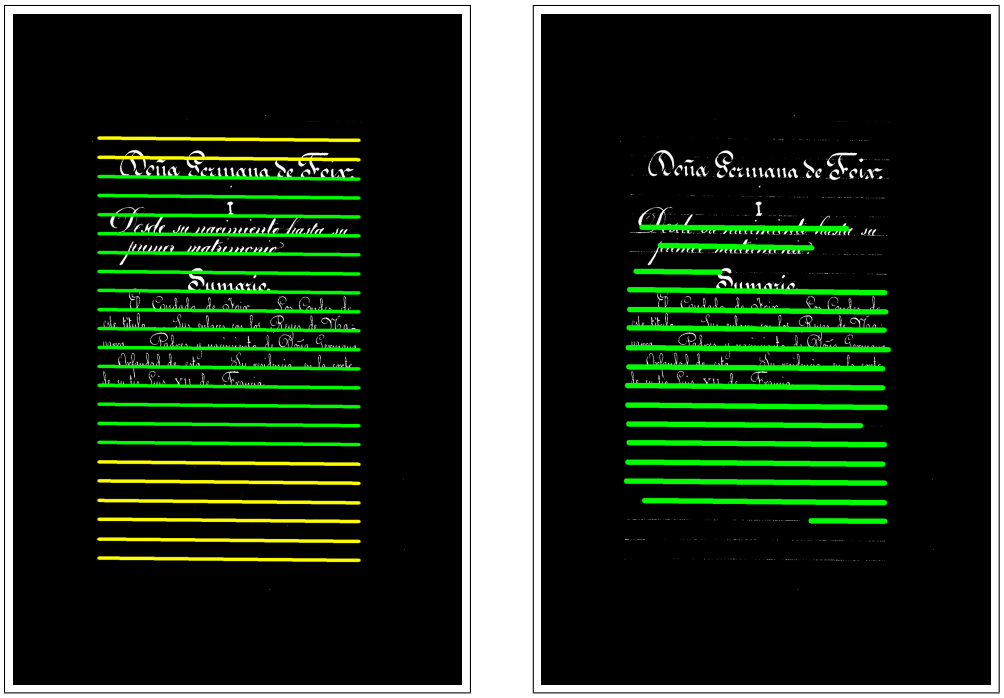


(a) Detected clusters. (b) Intermediate lines. (c) Clusters of line segments. (d) Final results.

Figure 2: Snapshots for the intermediate results generated by our model-based ruling line detection algorithm on a Germana sample. Note that in (d), there is a missing line detected at the top.

Table II: Performance of our model-based ruling line detection and its comparison with one algorithm in the literature.

Error Statistics	**X-position $\mathcal{P}.x$**	**Y-position $\mathcal{P}.y$**	**Length $\mathcal{L}$**	**# of lines $\mathcal{K}$**	**Spacing β_1**	**Skew β_2**	**Thickness $\mathcal{H}$**
Synthesis Dataset							
Mean (μ)	-4.16	0.55	6.9	0	0	-0.01	0
Standard Deviation (σ)	5.60	2.18	5.11	0	0.01	0.05	1.41
Germana Dataset							
Mean (μ)	-12.30	3.80	12.85	0.25	-0.49	-0.09	0.25
Standard Deviation (σ)	39.01	20.70	34.03	1.25	2.24	0.26	0.22
Germana Dataset (Zheng, *et al.* [4])							
Mean (μ)	-49.40	57.60	70.40	-1.20	-2.40	0.02	0
Standard Deviation (σ)	62.64	113.30	62.12	5.68	8.82	1.09	0



(a) Our results on a Germana sample. (b) Zheng, *et al.*'s results.

Figure 3: Comparison with Zheng, *et al.*'s algorithm [4].