LAMP-TR-109
CAR-TR–994
CS-TR-4545
UMIACS-TR-2003-113

December 2003

A PARALLEL LINE DETECTION ALGORITHM BASED ON HMM DECODING

Yefeng Zheng
Huiping Li
David Doermann

Language and Media Processing Laboratory
Institute for Advanced Computer Studies
University of Maryland
College Park, MD 20742-3275
(zhengyf,huiping,doermann)@cfar.umd.edu

Abstract

The detection of groups of parallel lines is important in applications such as form processing and text (handwriting) extraction in rule lined paper. These tasks can be very challenging in degraded documents where the lines are severely broken. In this paper we propose a novel model-based method which incorporates high level context to detect these lines. After preprocessing and skew correction, we use trained Hidden Markov Models (HMM) to locate the optimal positions of all lines simultaneously, based on the Viterbi decoding. The algorithm is trainable, therefore, it can easily be adapted to different application scenarios. The experiments conducted on known form processing and rule line detection show our method is robust, and achieved better results than other widely used line detection methods, such as the Hough transform, projection or vectorization-based methods.

Keywords: Line Detection, Form Processing, Hidden Markov Model, Document Image Analysis

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

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE **DEC 2003**	2. REPORT TYPE	3. DATES COVERED **00-12-2003 to 00-12-2003**

4. TITLE AND SUBTITLE **A Parallel Line Detection Algorithm Based on HMM Decoding**	5a. CONTRACT NUMBER 5b. GRANT NUMBER 5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER 5e. TASK NUMBER 5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Language and Media Processing Laboratory,Institute for Advanced Computer Studies,University of Maryland,College Park,MD,20742-3275**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR’S ACRONYM(S) 11. SPONSOR/MONITOR’S REPORT NUMBER(S)

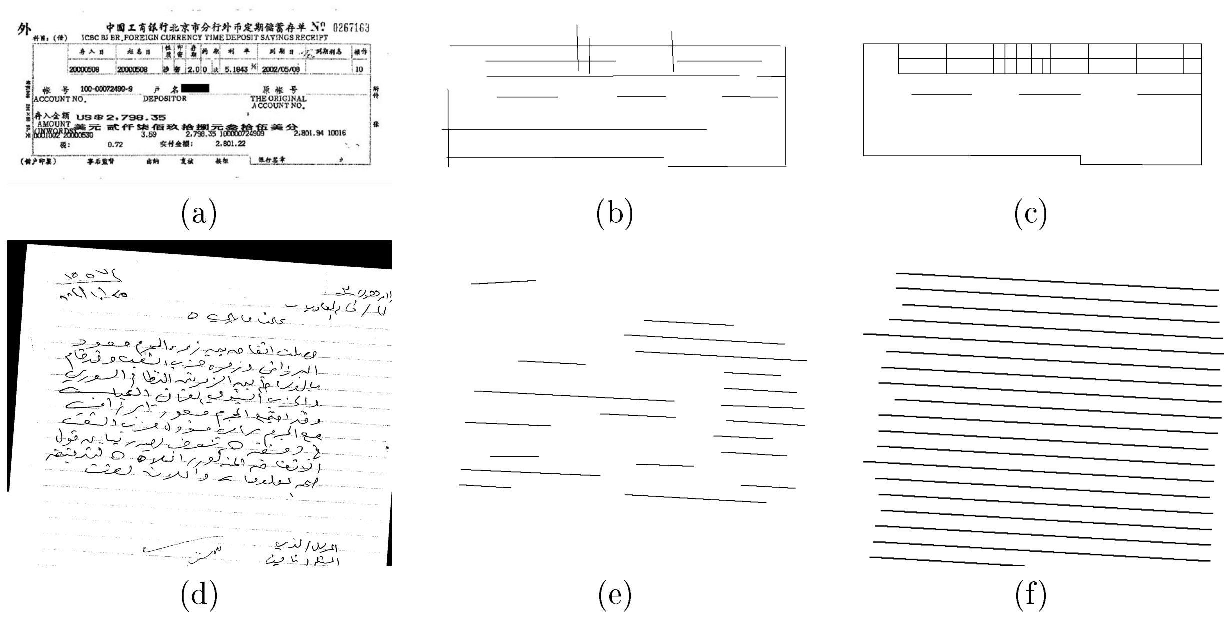
12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
The original document contains color images.

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**		**28**	


(a) (b) (c)

(d) (e) (f)

Figure 1: Line detection results for (a) a bank deposit form and (d) a rule lined document; (b) and (e) are line detection results using the DSCC method; (c) and (f) are our model based line detection results.

1 Introduction

The detection of groups of parallel lines is of particular interest in some applications. Fig. 1a shows an example of a form document with a group of horizontal and vertical parallel lines respectively. These lines provide crucial information for model-based form analysis and processing, such as form identification, form registration, content understanding, etc. Another example is a rule lined document shown in Fig. 1d. These lines are originally printed on the paper to guide users' writing. After digitization they will, however, touch text and cause problems for further document analysis such as segmentation and recognition. It is important that those lines can be detected and removed before we feed the text to an Optical Character Recognition (OCR) engine.

Many line detection algorithms have been presented in the literature [2,5,13,21,33,35]. They can work well on relatively clean documents with solid or mildly broken lines, but the performance will be significantly deteriorated if lines are severely broken due to the low image quality, or mix, touch, and/or overlap with text. Figs. 1b and 1e show the DSCC (Directional Singly Connected Chain) based line detection results for the form and rule lined documents. We can see only a few lines are partially detected in both cases due to severe brokenness, and a long horizontal line is falsely detected on the form document (Fig. 1b) since several horizontal strokes of Chinese characters happen to be close enough and lie on the same line. It is very difficult, if not impossible, to reliably detect these lines individually.

To handle these problems the context is often required to refine the initial detection. For example, in form processing most form cells are rectangular; in known form processing the number of lines and the gaps among these lines can be used as a priori knowledge and stored as references in form templates. These ideas have been presented in previous work to improve detection accuracy and reduce false alarms [1,9,30,35]. But the usage of the priori knowledge in the above applications is in an ad hoc way and lacks a systematic representation.

In this paper we propose a model-based method which incorporates context to optimally detect parallel lines systematically. Under the model, lines are detected by a Hidden Markov Model (HMM) decoding process which can determine the positions of all lines simultaneously. Rather than detecting lines directly on original images [2, 21, 35], we use a DSCC-based scheme to filter text as a preprocessing step so the interference with text can be minimized. We then use a coarse-to-fine approach to estimate the skew angle of the document. After deskewing the document we perform horizontal and/or vertical projections. Rather than simply treating the peaks of the projection profile as the line positions [2, 21], we model the projection profile with a HMM model so the context among these lines can be incorporated. The Viterbi algorithm is then used to search the optimal positions of these lines simultaneously from the projection profile. The model is trainable, therefore, it can be easily adapted for different applications. Two different detection scenarios (known form processing and rule line detection) presented in the paper illustrate the strength and advantage of our method. The experimental results show our method is robust, and can detect lines with high accuracy and low false alarms in degraded documents.

The remainder of this paper is organized as follows. In the next section we briefly survey the literature of line detection and form processing, followed by preprocessing in Section 3. Section 4 presents our HMM-based line detection algorithm in detail. Two application scenarios, known form processing and rule line detection, are described in Section 5. We demonstrate the robustness of our method with quantitative evaluation experiments in Section 6. The paper concludes with a brief summary and a discussion of the future work.

2 Related Work

2.1 Line Detection

Line detection is widely used in table detection and interpretation [2, 33, 35], engineering graph interpretation [4], and bank check/invoice processing [1, 30]. The algorithms presented can be broadly divided into two types: Hough transform based and vectorization based [24]. The Hough transform converts the global pattern detection problem in an image space to a local pattern (ideally a point) detection problem in a transformed parameter space [11, 13]. To detect a straight line, each black pixel (x, y) in an image space is transformed into a sinusoidal curve in the Hough parameter space

$$\rho = x\text{cos}\theta + y\text{sin}\theta \tag{1}$$

After transformation, collinear points (x_i, y_i) in the image space intersect at a point (ρ, θ) in the Hough parameter space. Therefore, a peak in the transformed space provides strong evidence that a corresponding straight line exists in the image. The Hough transform can detect dashed and mildly broken lines. However, it is very time consuming. To reduce the computational cost, a projection based method is proposed in [21] to detect form frame lines by limiting the search orientations since usually only horizontal and/or vertical lines exist in form documents. The method deskews the document first, and then detects the peaks of the horizontal and vertical projection profiles as lines. It can be viewed as a special case of the Hough transform by searching θ only around 0^o and 90^o. The method will fail if the projection of a line does not form a peak on the profile when it mixes with text, or the estimated skew angle is not accurate enough, or the lines are too short or severely broken. Chen and Lee proposed the strip projection method to alleviate this problem since lines are more likely to form peaks on the projection profile in a small region [2]. For horizontal line detection, they first divide an image into several vertical strips with equal width, and then perform horizontal projection in each strip. The detected co-linear line segments in each strip are linked to form a whole line.

Vectorization based line detection algorithms extract vectors from an image first, and then merge vectors into lines [3, 24]. A vector can be defined as a collection of pixels which satisfy the pre-defined criterion. It can be a chain of pixels, a line segment, or an arc. Using vectors instead of pixels can significantly reduce the number of elements to be processed, typically from 100:1 to 10:1. The major difference among existing vectorization based algorithms is the definition and extraction of vectors, while the merging process is often rule-based and very similar for different algorithms.

Thinning is a common method to extract vectors. It uses an iterative boundary erosion process to remove outer pixels until only a skeleton of pixel chains remains [29]. It can maintain connectivity, but also tends to create noisy junctions at corners, intersections, and branches. Medial line methods, on the other hand, extract image contours as pixel chains first. Then the middle points of the perpendicular lines, which project from one side of the contour to the other, form a medial line [14]. The methods may miss pairs of contour lines at branches, so postprocessing is often required to reduce this distortion [10]. The result of either thinning or medial line methods is a chain of pixels, and a line can be detected by approximating the pixel chain. Recently, the *Sparse Pixel Vectorization* (SPV) algorithm, proposed by Dori et al. [5], does not use contours to get medial lines. It traces the medial axis points of consecutive horizontal or vertical pixel runs until some constraints are violated. Each continuous trace is a vector, representing a bar or an arc. SPV often achieves better vectorization results than other medial line methods, but the vector extraction is complicate, and often needs postprocessing to refine the results.

As a run-length based approach, *Block Adjacency Graph* (BAG) is a generic structure to represent an image [33, 34]. BAG is defined as $\mathcal{G}(N, E)$, where N is a set of block nodes and E is a set of edges indicating the connection between two nodes. Each node is a block which contains either one or several horizontal run lengths adjacently connected in vertical direction and aligned on both left and right sides within a given tolerance. A Line is detected by searching a connected sub-graph in the BAG with large longitude. As a generic image presentation method, a BAG does not consider the special characteristics of a straight line. Therefore, it can only detect unbroken lines within a small skew range ($< 5^o$).

Recently Zheng et al. presented a vectorization based algorithm called Directional Singly Connected Chain (DSCC) method [35]. A DSCC is a chain of run-lengths which are singly connected. Each DSCC represents a line segment and multiple non-overlapped DSCCs are merged into a line, based on pre-defined rules. The basic characteristic of a line is that it has only one running direction. When a junction is encountered, the merging process stops and a new DSCC is generated.

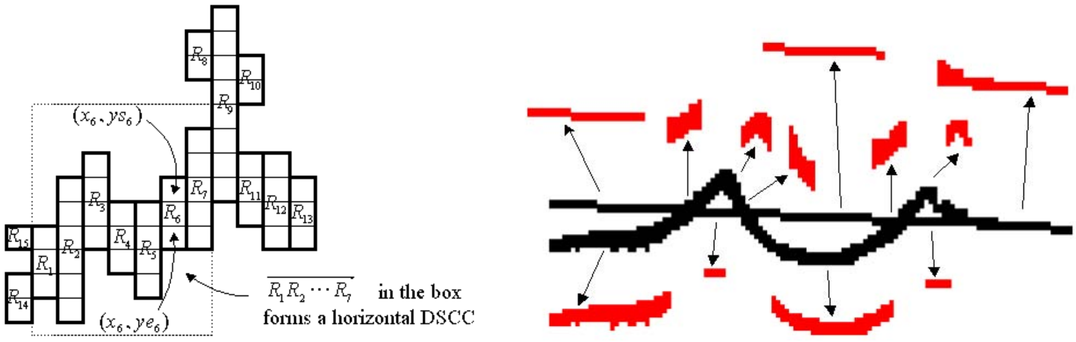
2.2 Model-Based Form Processing

Millions of form documents, such as health insurance forms, checks, and bank slips, are being processed everyday. Some systems have been developed to process these forms automatically or semi-automatically [1, 8, 30]. Form processing can be categorized as *unknown* and *known* form processing [1]. Unknown form processing assumes no a priori knowledge from the input forms, and extracts all information based on low level image analysis. Errors are often expected and user assistance is required. Known form processing, on the other hand, is designed to process a pre-defined set of forms, where a priori information can be stored as templates in the database to guide the later processing. It is widely used in banks, post offices, and tax offices where the types of forms can be predefined. For an input form, the system first selects the template which matches it best (*form identification*). Then some anchors (such as specific marks, form frame lines, etc.) are detected for the registration so the variations produced by scanning (e.g. rotation, translation, and scaling) can be compensated. Finally the identified template is used to guide the system to recognize interested fields on the form (different OCR engines can be used for different fields), and output the recognition results to a database. Though special anchors may be available to facilitate the form identification and registration for specially designed forms, more general approaches use features related to frame lines explicitly or implicitly, such as frame lines [1,30–32], form cells [22], and the cross points of frame lines [6], etc., for form identification and registration. Robust detection of frame lines is crucial in these approaches.

For line-based image registration, model-based approaches are often used to find the correspondence between the detected lines and those stored in the form template [1,8,30]. In [30], Tang and Suen first detect an anchor line, from which the approximate locations of other lines can be determined based on the form template. Then the system refines the detection by searching in areas around these locations. The correspondence between the detected anchor line and those in the form template may not be unique. Cesarini et al. proposed a hypothesis and verification paradigm to solve the correspondence and detection simultaneously [1]. A hypothesis is generated for each correspondence, and then the system searches the expected lines in the surrounding areas to verify the hypothesis. One shortcoming of the approach is that the hypothesis and verification need to be defined for different types of forms. The false alarms are reduced due to the use of templates, but miss detection of broken lines is reported [30].

3 Preprocessing

The purpose of preprocessing is two-fold: first we filter out text strokes so their effect can be diminished; second, we deskew the document so the parallel lines can be oriented horizontally or vertically.



(a) (b)

Figure 2: The definition of a horizontal DSCC. (a) An example of DSCCs; (b) extracted DSCCs (represented in red color) where a text stroke crosses a line.

3.1 Text Filtering Based on DSCC

3.1.1 Definition of DSCC

We define two types of DSCCs: the horizontal one and the vertical one, which are used to detect horizontal and vertical frame lines respectively in [35]. Take the horizontal DSCC for example: a horizontal DSCC C_h consists of a black pixel run-length array $\overline{R_1 R_2 \cdots R_m}$, where R_i is a vertical run-length with one pixel width

$$R_i(x_i, ys_i, ye_i) = \left\{ (x, y) \;\middle|\; \begin{array}{l} p(x,y) = 1, \text{for } x = x_i, y \in [ys_i, ye_i] \\ \text{and } p(x_i, ys_i - 1) = p(x_i, ye_i + 1) = 0 \end{array} \right\} \tag{2}$$

where $p(x, y)$ is the value of pixel (x, y) with 1 representing black pixels, and 0 representing white pixels; x_i, ys_i and ye_i designate x, starting y, and ending y coordinates of R_i respectively. Two neighboring run-lengths R_i and R_{i+1} are merged into a DSCC chain if they are singly connected in the horizontal direction. As shown in Fig. 2, the single connection means that at each side of $R_i (1 < i < m)$, there is one and only one connected run-length. In this example, $\overline{R_1 R_2 \cdots R_7}$, $\overline{R_{11} R_{12} R_{13}}$, R_8, R_9, R_{10}, R_{14} and R_{15} are extracted as DSCCs. The definition of vertical singly connected chain C_v is very similar to the horizontal one.

The most important property of a line is the single connection along its running direction. An ideal line consists of only one DSCC. A real line often consists of multiple non-overlapping DSCCs. Fig. 2b shows an example of extracted DSCCs (represented in red color) of a text stroke crossing a line. We can see that the line is broken into several line segments (DSCCs) on the touching area. If the image quality is reasonable, then a line can be detected by merging those DSCCs with the similar orientation [35]. In our case we use it to remove text and preserve line segments.

3.1.2 Filtering

As shown in Fig. 2b, a DSCC can be a text stroke or a line segment. We observed that a line segment often has a smaller variation from the desired orientation and larger aspect

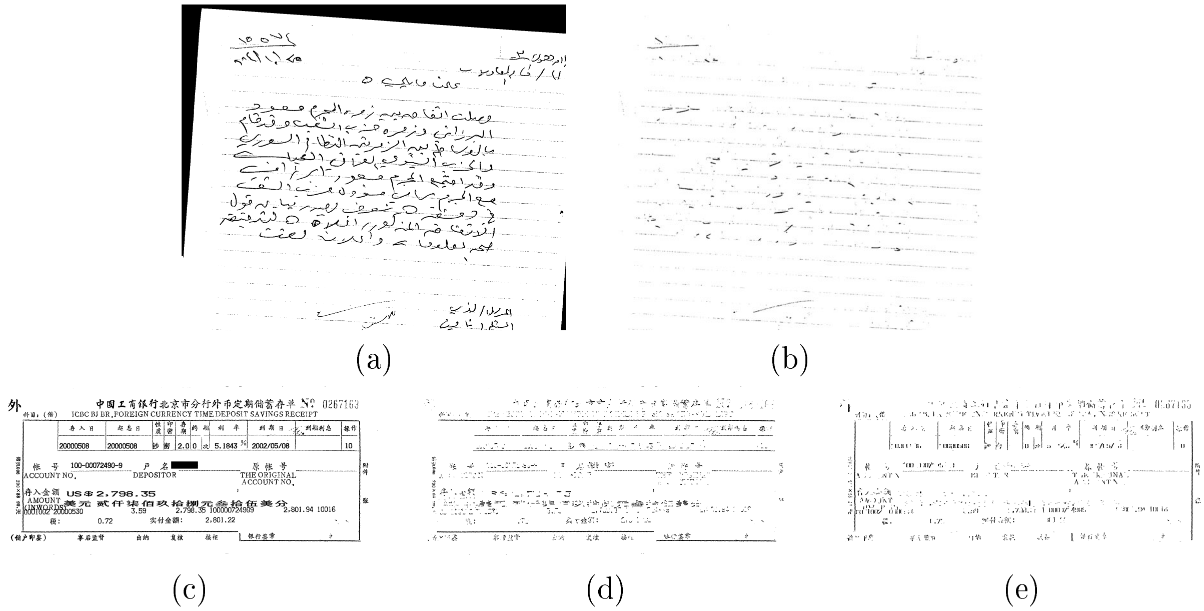


Figure 3: Text filtering using DSCC. (a) A document image with rule lines; (b) result of text filtering in horizontal direction; (c) a form document image; (d) result of text filtering in horizontal direction; (e) result of text filtering in vertical direction.

ratio. We use an ellipse to model the shape of a DSCC, and calculate the orientation θ, the first and second axes a and b of each DSCC as follows

$$\mu_{pq} = \sum_{x}\sum_{y}(x-\bar{x})^p(y-\bar{y})^q \mathrm{I}(x,y) \tag{3}$$

$$\theta = 0.5\tan^{-1}\left(\frac{2\mu_{11}}{\mu_{20}-\mu_{02}}\right) \tag{4}$$

$$a = \sqrt{\frac{2[\mu_{20}+\mu_{02}+\sqrt{(\mu_{20}-\mu_{02})^2+4\mu_{11}^2}]}{\mu_{00}}} \tag{5}$$

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

where $I(x,y)$ represents a pixel in the DSCC, $\bar{x}$ and $\bar{y}$ are the means of x and y coordinates, and u_{pq} is the central moment. For horizontal line detection, we only preserve those DSCCs with either very small sizes ($\max\{a,b\} < T_1$), or large aspect ratios within a specified orientation angle ($a/b > T_2$ and $\theta \in [-45^o, 45^o]$). T_1 and T_2 are thresholds determined experimentally. The first condition preserves small DSCCs which can be parts of a broken line, or the touching areas of lines and text; and the second one preserves large DSCCs which are likely to be horizontal line segments. Similar conditions hold for vertical lines except for the orientation angle. Fig. 3 shows some examples of text filtering. We can see that most text strokes are filtered and the line segments are preserved.

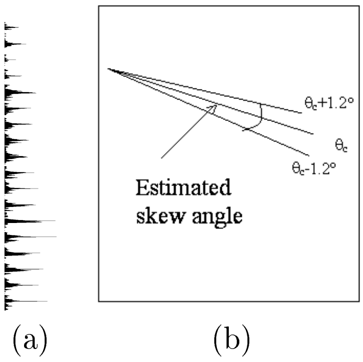


Figure 4: Refinement of the skew angle estimation. (a) Horizontal projection along the skew angle; (b) search range.

3.2 Skew Angle Estimation

Our skew angle estimation is based on the line segments extracted. After filtering text strokes, we merge the preserved DSCCs into longer line segments and use a two-step coarse-to-fine method to estimate the skew angle. In the coarse estimation, we construct a weighted angle histogram of all extracted lines as follows

```
Clear all entries of the histogram $h$
For each line $i$ do
    $\theta_i$ = orientation of line $i$
    Weight $w_i$ = length of line $i$
    $h(\theta_i) = h(\theta_i) + w_i$
End
```

The length of a 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 estimate further by performing projection along the angles within a small range around θ_c, as shown in Fig. 4. The angle $\hat{\theta}$ maximizing the variance of the projection is the refined estimate

$$\hat{\theta} = arg \max_{\theta\in[\theta_c-1.2^o,\theta_c+1.2^o]} Var(h(y,\theta)) \tag{7}$$

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. After skew angle estimation, the skew distortion is corrected before we perform horizontal or vertical projection.

4 HMM Model-Based Parallel Line Detection

In the following description, we use horizontal line detection as an example to illustrate the proposed method. The extension to vertical line detection is straightforward. A stochastic model, $\mathcal{M}(y_1, y_2, \ldots, y_N)$, is proposed for a group of parallel lines, where N is the number of lines, and $y_i, i = 1, 2, \ldots, N$ is the vertical position of the i^{th} line. The line gap g_i between two neighboring lines i and $i+1$ is defined as

$$g_i = y_{i+1} - y_i \tag{8}$$

As we show later, the line positions $\{y_i, i = 1, 2, \ldots, N\}$ form a Markov chain if the variations in line gaps are independent. As they are not observable in the projection profile, a HMM model is more suitable for modeling the profile. The line positions can be detected by decoding the HMM model.

4.1 Markov Chain

The Markov property of a sequence of events is well studied in the literature [27]. Consider a system that stays at one of a set of N distinct states, $S_1, S_2, \ldots, S_N$, at any sampling time t. It undergoes a change of state according to a set of probabilities associated with the state during the period between two successive sampling times. For a Markov chain (the first order), the probability of staying at state q_t only depends on the previous state q_{t-1}

$$P[q_t = S_{k_t} | q_{t-1} = S_{k_{t-1}}, q_{t-2} = S_{k_{t-2}}, \ldots, q_1 = S_{k_1}] = P[q_t = S_{k_t} | q_{t-1} = S_{k_{t-1}}] \tag{9}$$

If the state transition probability is independent of time t, then the Markov chain is said to be *homogeneous*

$$P[q_t = S_j | q_{t-1} = S_i] = a_{ij} \quad 1 \leq i, j \leq N \tag{10}$$

We can show that line positions $\{Y_i, i = 1, 2, \ldots, N\}$ form a Markov chain, if the variations in line gaps are independent. Here we use uppercase characters to represent random variables (e.g. Y_i) and lowercase characters to represent the value of the random variables (e.g. y_i).

Theorem 1: Let $Y_i, i = 1, 2, \ldots, N$ be line positions, and $G_i = Y_{i+1} - Y_i, i = 1, \ldots, N-1$ be line gaps. If $\{G_i\}$ are independent, then $\{Y_i\}$ form a Markov chain.

$$P(Y_i | Y_1, Y_2, \ldots, Y_{i-1}) = P(Y_i | Y_{i-1}) \tag{11}$$

Proof:

$$\begin{aligned} P(Y_i | Y_1, Y_2, \ldots, Y_{i-1}) &= P(G_{i-1} + Y_{i-1} | Y_1, Y_2, \ldots, Y_{i-1}) \\ &= P(G_{i-1} | Y_1, Y_2, \ldots, Y_{i-1}) \\ &= P(G_{i-1} | G_1, G_2, \ldots, G_{i-2}, Y_{i-1}) \end{aligned} \tag{12}$$

Since, $\{G_i\}$ are independent, we have

$$\begin{aligned} P(Y_i | Y_1, Y_2, \ldots, Y_{i-1}) &= P(G_{i-1} | Y_{i-1}) \\ &= P(G_{i-1} + Y_{i-1} | Y_{i-1}) \\ &= P(Y_i | Y_{i-1}) \end{aligned} \tag{13}$$

Therefore, $\{Y_i\}$ form a Markov chain.

In random process, $\{Y_i\}$ is called an independent increment process, which includes several well known random processes, e.g. Brownian motion process, Poisson process, etc [7].

4.2 HMM Model

In many applications, the actual state sequence is not observable. The resulting model (which is called a hidden Markov model) is a doubly embedded stochastic process with an underlying stochastic process that is not observable, but can only be observed through another stochastic process that produces the sequence of observations. The elements of a standard discrete HMM model are

1) N, the number of the states in the model.
2) M, the number of distinct observation symbols per state.
3) $A = \{a_{ij}\}$, the state transition probability matrix.
4) $B = \{b_{ij}\}$, the probability distribution matrix of the observation symbols.
5) π, the initial state distribution.

HMM models can model some 1-D signals well, and have achieved great success in speech [27] and handwriting recognition [26].

In our application, we can only observe the projection profile h_k

$$P(H_k = h_k | Y_1 = y_1, \ldots, Y_N = y_N) = \begin{cases} P(H_k = h_k | \exists i, k = y_i) & \text{A line is on } k \\ P(H_k = h_k | \forall i, k \neq y_i) & \text{No lines are on } k \end{cases} \tag{14}$$

Therefore, the projection profile can be modeled with a HMM model. A standard HMM model is shown in Fig. 5a, where S_T and S_B are the states representing top and bottom image borders, $S_{L,i}, i = 1, 2, \ldots, N$ represent lines, and $S_{G,i}, i = 1, 2, \ldots, N-1$ represent the line gaps between lines i and $i+1$.

One weakness of conventional HMMs is the modeling of state duration. The inherent duration probability distribution $p_i(d), d = 1, 2, \ldots,$ associated with state S_{G_i} is

$$p_i(d) = (a_{ii})^{d-1}(1 - a_{ii}) \tag{15}$$

where a_{ii} is a self transition probability. The exponential state duration distribution is inappropriate for our applications. Instead we explicitly model the duration distributions. The model with explicit state duration is shown in Fig. 5b, where the stochastic property of the model is incorporated into the state duration distributions $P_T(d), P_B(d), P_i(d), i = 1, 2, \ldots, N-1$. For some applications, the quality of the modeling is significantly improved when explicit state duration distributions are used [19].

4.3 HMM Model Parameter Estimation

The major drawback of an explicit duration HMM model is that it significantly increases computational costs for model training. With a traditional forward-backward training algorithm (a type of EM algorithm), the re-estimation problem for a variable duration HMM is more difficult than that for a standard HMM [27]. Fortunately, in our case, we can directly get the HMM parameters from groundtruth since the states explicitly correspond to image components. Therefore, the forward-backward training algorithm is not needed. We set duration probabilities of states S_T and S_B to uniform distribution within a range. The duration probabilities of states $S_{G,i}, i = 1, 2, \ldots, N-1$ are estimated directly from the groundtruth.

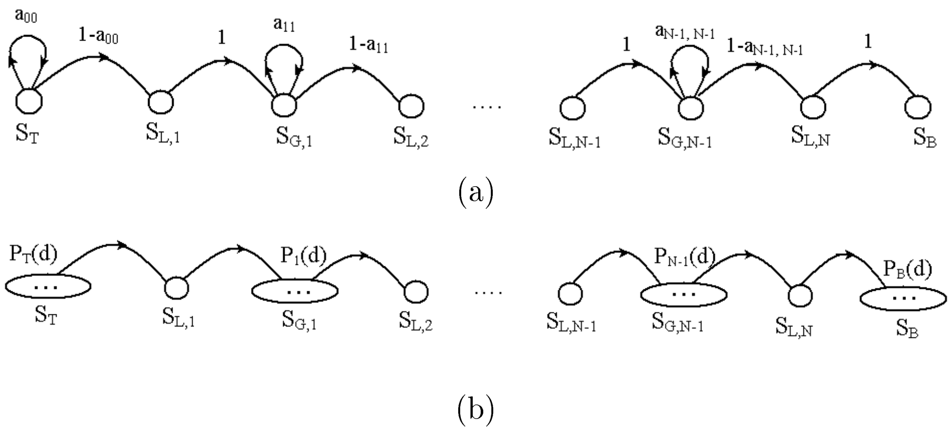


Figure 5: HMM models for a projection profile. (a) A standard HMM model; (b) a HMM model with explicit state duration.

The observation comes from the projection profile $\{h_k\}$. The large number of observation symbols would prevent us from estimating the model parameters reliably with limited training samples. There are two methods to reduce the number of parameters of the model. One is to model the distribution of the observation as a Gaussian distribution [27], so only the mean and variance of the Gaussian distribution are needed to be estimated. For known form processing, we find the projections of a line over multiple form instances can be well modeled as a Gaussian distribution. Another method quantizes the projection profile into several levels. For rule line detection, the image quality varies significantly among different images. The distribution of the observation does not follow the Gauss distribution. Therefore, we quantize h_k into K levels ($K = 5$ in our experiments for rule line detection). The probabilities of each level can be estimated from the groundtruth.

The HMM parameters estimated directly from the groundtruthed data set are not optimal due to the sparseness of the training data. For example, some entries of the line gap distribution in Table 1 do not appear or only appear a few times. Parameter sharing, a technique used in Neural Network to train the parameters with limited training samples [18, 28], is used in our approach. For example, we let non-line states $S_T, S_B, S_{G,1}, \ldots, S_{G,N-1}$ share the same observation probability distributions since the observations of these states are the same: the projections of noise and remaining text strokes after filtering. For rule line detection, we further combine all line states into one state, and all non-line states into another state, which significantly reduces the parameters of the model. For line gap distribution estimation, we assume the distribution is symmetric around the mean value. Therefore, data smoothing techniques, originally proposed in natural language processing [15], can be used

$$C'(\bar{g}_i + k) = C'(\bar{g}_i - k) = \frac{C(\bar{g}_i + k) + C(\bar{g}_i - k)}{2} \quad k = 1, 2, \ldots \tag{16}$$

where $\bar{g}_i$ is the mean value of line gap G_i, $C(k)$ is the number of instances of G_i with value k in the training set, and $C'(k)$ is the smoothed result after imposing symmetric regularization. Finally, we set the empty entries to the minimal value of all non-zero entries. Suppose the maximal variation of the line gap G_i is K. For $k \in [-K, K]$, the final smoothed result is

$$C''(\bar{g}_i + k) = \begin{cases} C'(\bar{g}_i + k) & \text{if } C'(\bar{g}_i + k) \neq 0 \\ \min_{i \in [-K,K], C'(\bar{g}_i + i) \neq 0} C'(\bar{g}_i + i) & \text{if } C'(\bar{g}_i + k) = 0 \end{cases} \tag{17}$$

$C''(k)$ can be converted to probability by normalization. We will illustrate the data smoothing with examples in the following section.

The ultimate goal of training is to search the optimal HMM model parameters to minimize the line detection error. The estimated parameters from the training data can produce reasonable results, however they do not minimize the line detection error rate. Generally the error criterion is a very complex function of the model parameters without a closed form. A direct searching algorithm can be used to solve such optimization problems. In our case, the simplex search method proposed by Nelder and Mead is used to minimize the detection error [25].

4.4 Decoding of the HMM Model

Given the observation sequence $O = h_k, k = 1, 2, \ldots, T$, and the HMM model λ, we want to search an optimal state sequence $Q = q_1 q_2 \ldots q_T$ to maximize $P(Q|O, \lambda)$, which is equivalent to maximizing $P(Q, O|\lambda)$. Normally, the Viterbi algorithm, a dynamic programming method, is used to decode HMM models. A matrix v with dimension $T \times (N + 1)$ is defined and updated in the Viterbi algorithm, and

$$v(t, n) = \max_{q_1, q_2, \ldots, q_{t-1}} P[q_1, q_2, \ldots, q_t = S_{L,n}, h_1, h_2, \ldots, h_t|\lambda] \tag{18}$$

is the best decoding score at time t, which accounts for the first t observations and ends in state $S_{L,n}$. The sequence $q_1, q_2, \ldots, q_{t-1}$ maximizing the probability in Eq. (18) is the best decoding result until time t if we decode state q_t as the n^{th} line.

Suppose the minimal and maximal state durations of states $S_{G,n}$ are δ_{n-} and δ_{n+}, and the durations of S_T and S_B are uniformly distributed in $[0, \delta_T]$ and $[0, \delta_B]$, respectively. The complete procedure of decoding is stated as follows

1. Clear all entries of matrix v.

2. For $1 \leq i \leq \delta_T$, decode the first $i - 1$ observations as S_T (the top image border) and observation i as $S_{L,1}$

$$v(i, 1) = \frac{1}{\delta_T + 1} P(h_i|q_i = S_{L,1}) \prod_{j=1}^{i-1} P(h_j|q_j = S_T), \tag{19}$$

where $P(h_i|q_i = S_{L,1})$ is the probability of observing h_i if the system enters state $S_{L,1}$ at time i; $\prod_{j=1}^{i-1} P(h_j|q_j = S_T)$ is the probability of observing the first $i - 1$ observations if the system stays at state S_T during the time period from 1 to $i - 1$; and $\frac{1}{\delta_T+1}$ is the probability of the model staying at S_T for $i - 1$ consecutive periods.

3. Set $t = 1$.

4. For $n = 1$ to N
   For $j = \delta_{n-}$ to δ_{n+}

$$\begin{aligned} v'(t+j,n) &= v(t,n)P_n(j)P(h_{t+j}|q_{t+j} = S_{L,n+1}) \prod_{k=t+1}^{t+j-1} P(h_k|q_k = S_{G,n}) && (20) \\ v(t+j,n) &= \max\{v(t+j,n),\ v'(t+j,n)\} && (21) \end{aligned}$$

   End loop of j
   End loop of n
   Here $P_n(j)$ is the probability of staying at state $S_{G,n}$ with j consecutive times; $P(h_{t+j}|q_{t+j} = S_{L,n+1})$ is the probability of observing observation h_{t+j} if the system enters $S_{L,n+1}$ at time $t+j$, which corresponds to a new line; and $\prod_{k=t+1}^{t+j-1} P(h_k|q_k = S_{G,n})$ is the probability of observing sequence h_{t+1} to h_{t+j-1} if the system stays at state $S_{G,n}$ during this time period, which corresponds to a line gap. Eq. (21) updates the optimal partial detection result.

5. If $t > T - \delta_B$, decode the following sequence as the bottom image border.

$$\begin{aligned} v'(T,N+1) &= v(t,N)\frac{1}{\delta_B+1} \prod_{k=t+1}^{T} P(h_k|q_k = S_B) && (22) \\ v(T,N+1) &= \max\{v(T,N+1),\ v'(T,N+1)\} && (23) \end{aligned}$$

6. If $t < T$, then $t = t+1$, and go to step 4.

For each t, the algorithm remembers the best decoding path until time t. After decoding,

$$v(T,N+1) = \max_Q P(Q,O|\lambda) \quad (24)$$

is the probability of detecting lines given the model, which can be regarded as detection confidence. The sequence $q_1, q_2, \ldots, q_T$ which achieve $v(T,N+1)$ is the optimal decoding result.

4.5 Polyline Representation

After identifying the vertical position of a line, our next step is to detect the left and right end points by grouping the broken line segments together. For each detected line, those DSCCs within 10 pixels distance to the detected line are merged [35]. Those lines with less than 50 pixels are removed.

An ideal straight line can be represented with two parameters a and b as $y = a \times x + b$. Practically, a real line is represented with points $(x_i, y_i), i = 1, 2, \ldots, n$. The parameters a and b can be estimated based on the minimum mean squared error criterion (MMSE)

$$\bar{x} = \sum_{i=1}^{n} x_i/n \qquad \bar{y} = \sum_{i=1}^{n} y_i/n$$

$$a = \frac{\sum_{i=1}^{n}(x-\bar{x})(y-\bar{y})}{\sum_{i=1}^{n}(x-\bar{x})^2}$$
$$b = \bar{y} - a \times \bar{x} \tag{25}$$

For most straight lines, this approximation is good enough. However, due to the distortion introduced by photocopying and scanning, some lines are cursive, and can not be represented by two end points well. In this case, a polyline representation is used as follows

1. Calculate the average approximation error of a line

$$\delta y_i = |y_i - a \times x_i - b| \tag{26}$$
$$e = \sum_{i=1}^{n} \delta y_i / n \tag{27}$$

2. If e is smaller than the average line width (often 2-4 pixels), keep it with two end points representation, and exit.

3. Otherwise, split the whole line into two segments from the middle, and estimate the line parameters a and b for each segment respectively, as described in Eq. (25).

4. For each segment, go to step 1 and repeat.

A polyline is described as a sequence of vertices $(P_1, P_2, \ldots, P_m)$. Two or three segments are sufficient to represent most lines in our following experiments.

5 Applications

5.1 Known Form Processing

The application of the algorithm to known form processing is straightforward. Generally there are a group of horizontal and vertical parallel lines respectively on a form. Therefore, we use two HMM models to detect the horizontal and vertical lines separately. To apply the algorithm, we need to estimate two sets of parameters: 1) The distribution of the observation symbols of each state (B matrix in Section 4.2); and 2) The state duration probabilities of each state.

5.1.1 Estimation of the Distributions of Observation Symbols

In our case, the observation symbols are the projection profile, which is in the range of $[0, w]$ (where w is the width of the image). As we addressed in Section 4.3, a large number of observation symbols would cause difficulties in estimating the distributions reliably with limited training samples. Under some assumptions, we can show that using the Gaussian distribution to model projections of a line over multiple form instances is appropriate. In stochastic document image degradation models, a white (black) pixel is randomly selected and flipped to black (white) [16]. The projection is the summation of all black pixels on the line

$$h = \sum_{i=1}^{N} a_i \tag{28}$$

where

$$a_i = \begin{cases} 1 & \text{if black pixel } i \text{ is preserved} \\ 0 & \text{if black pixel } i \text{ is flipped to white during degradation} \end{cases} \tag{29}$$

Under white Gaussian noise, a widely used model for degradation, a_i follows a Bernoulli distribution: $a_i \sim Bernoulli(\rho)$, where ρ is the probability for a black pixel to be lost. Consequently, h follows a binomial distribution $Bin(\rho, N)$

$$P(h) = \binom{N}{h} \rho^{N-h}(1-\rho)^h \tag{30}$$

According to the central limit law, if N is large enough (or if the line is long enough), then the distribution of random variable h converges to a Gaussian distribution [7]

$$\lim_{N\to\infty} \frac{h - E[h]}{\sqrt{N\rho(1-\rho)}} \to \mathcal{N}(0,1) \text{ in distribution} \tag{31}$$

In known form processing, generally, a set of forms are captured with the similar imaging conditions. Therefore ρ is roughly consistent for each form instance in the set. So a Gaussian distribution is a good approximation for the projections of a line over multiple form instances. The mean and variance of the Gaussian distribution can be estimated from the groundtruth. Figs. 6a to 6f show the distributions of the projections of all six horizontal lines on a set of bank deposit forms with one instance shown in Fig. 1a. The histogram is generated over 100 form samples. We can see that the Gaussian distribution is a good approximation. For non-line states, the approximation is not good enough since a projection is always larger than zero, as shown in Fig. 6g. But we found in the experiments that the affect of this approximation error is negligible for the final line detection result.

5.1.2 Estimation of the State Durations

The state duration of $S_{G,i}, i = 1, 2, \ldots, N-1$, represents the line gap between lines i and $i+1$, which can be estimated from the groundtruth. Table 1 shows the distribution of the gap between the first and second horizontal lines on the bank deposit form in our database with 100 samples. The average value of the gap is 94 pixels. The row of *distance* lists the difference to the average value. The row of *raw occurrence* shows the number of occurrences the gap takes a specific value. We can see that the variation is from -9 pixels to 6 pixels, and the distribution is roughly symmetric around the average value. Due to the sparse data problem, some entries within the range of [-9, 6] are not observed in the training set, which will deteriorate the performance. Therefore, data smoothing is used. The row of *symmetric regularization* is the result after we impose the symmetry (Eq. (16)). At last, we set the zero entries to the minimal value of all non-zero entries, as

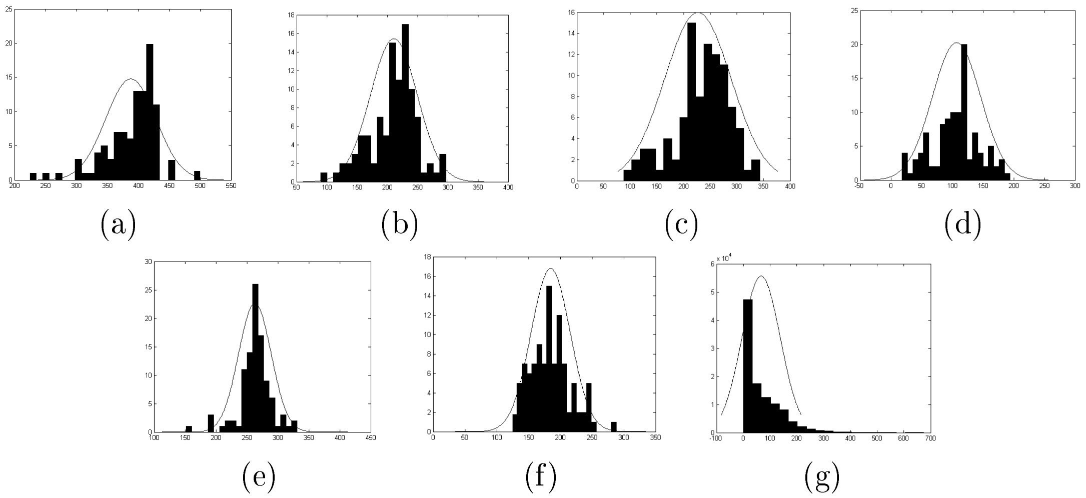


Figure 6: The distributions of the observation symbols (horizontal projections) for 100 scanned instances of a bank deposit form. One example of the form is shown in Fig. 1a. There are six horizontal lines on the form. (a) to (f) the histograms of the projections of six horizontal lines respectively; (g) the histogram of the projections of the non-line states.

Table 1: The distribution of the line gap between the first and second horizontal lines on a bank deposit form. The average is 94 pixels. The row of *distance* lists the difference to the average value.

Distance	-9	-8	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7	8	9
Raw Occurrence	1	0	2	0	0	3	5	12	18	24	16	6	8	4	0	1	0	0	0
Symmetric Regularization	.5	0	1	.5	0	3.5	6.5	9	17	24	17	9	6.5	3.5	0	.5	1	0	.5
Zero-Occurrence Smoothing	.5	.5	1	.5	.5	3.5	6.5	9	17	24	17	9	6.5	3.5	.5	.5	1	.5	.5

shown in the row of *zero-occurrence smoothing*. After data smoothing, the distribution $P_1(d)$ can be estimated by normalization. Similarly, we can get the distributions of other line gaps.

5.1.3 Decoding

After estimating parameters, we use the Viterbi algorithm described in Section 4.4 to decode the observation. Fig. 7 shows the decoding results of the Viterbi algorithm on the horizontal and vertical projection profiles of the bank deposit form (Fig. 1a). The locations picked up by the Viterbi algorithm are labeled with red squares. We can see that instead of picking the highest peaks as detected lines in the projection based methods [2, 21], our approach outputs the line positions which are most compatible with the model.

After detecting the horizontal and vertical lines, the method described in Section 4.5 can be used to determine the end points of the lines. However, if a line is severely degraded, the end points can not be determined accurately. In form processing, the cross points of horizontal and vertical lines can be used to determine the end points. Short lines

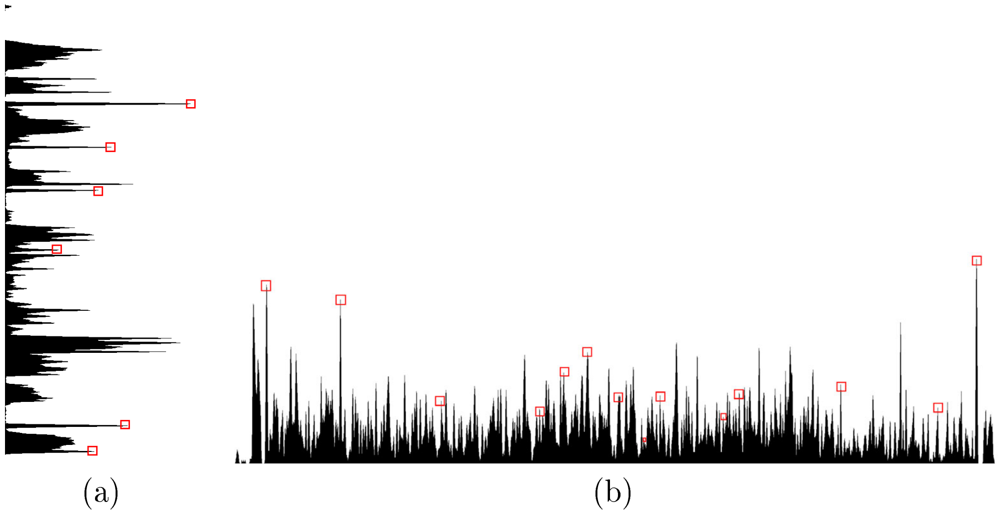


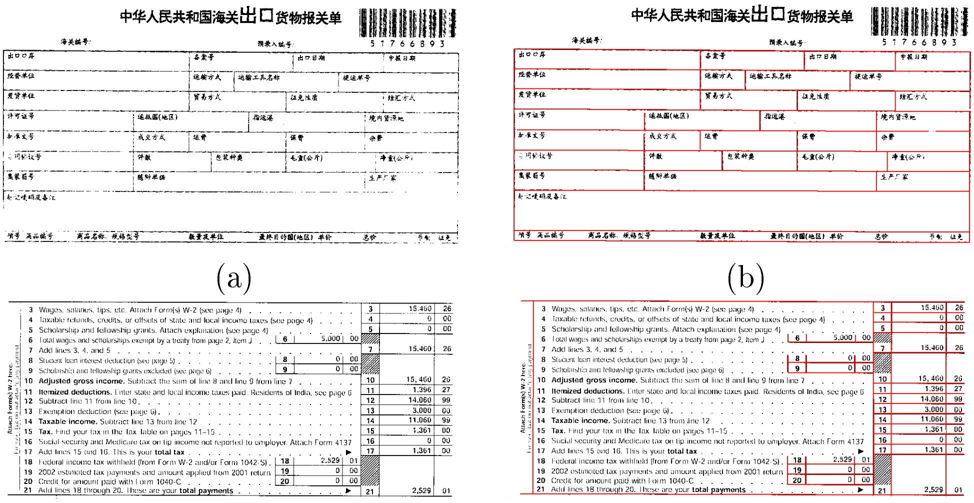
Figure 7: The lines detected after decoding the HMM models using the Viterbi algorithm on the horizontal (a) and vertical (b) projection profiles of the bank deposit form. The original form is shown in Fig. 1a. The locations picked up by the Viterbi algorithm are labeled with red squares. We can see that instead of picking the highest peaks as detected lines in the projection based methods, our approach outputs the line positions which are most compatible with the model.

lying on the same line (i.e., three dotted lines in the middle of the form, as shown in Fig. 1a) are merged as one line in the HMM model. After detecting all lines, each individual lines can be obtained using the geometric relationship of the end points to other lines. Fig. 1c shows the model based line detection result. We can see our method can even detect the short lines which do not form peaks on the projection profile (especially for the two shortest vertical lines) which are most likely missed by other methods, such as the Hough transform or projection based methods. Our method outputs the exact number of lines indicated by the model without false alarms. Fig. 8 shows two more examples of an export registration form used by the Customs Bureau of China and a portion of a U.S. income tax return form.

Another advantage of our HMM based form processing approach is that it can be extended for form identification easily. Suppose there are n form templates $\lambda_1, \lambda_2, \ldots, \lambda_n$. According to the Bayesian rule, $\hat{\lambda}$ which maximizes the posteriori probability is selected as the template for the input form

$$\hat{\lambda} = arg \max_{\lambda_i} P(\lambda_i|O) = arg \max_{\lambda_i} P(O|\lambda_i)P(\lambda_i) \qquad (32)$$

where O is the observation (the projection profile in our method), $P(\lambda_i)$ is the priori probability of form template λ_i, and $P(O|\lambda)$ is the probability of observing the sequence of observations given the model, which can be calculated efficiently with the forward algorithm [27].



中华人民共和国海关出口货物报关单

海关编号: 预录入编号: 5 1 7 6 6 8 9 3

出口口岸 | 备案号 | 出口日期 | 申报日期

经营单位 | 运输方式 | 运输工具名称 | 提运单号

发货单位 | 贸易方式 | 征免性质 | 结汇方式

许可证号 | 运抵国(地区) | 指运港 | 境内货源地

批准文号 | 成交方式 | 运费 | 保费 | 杂费

合同协议号 | 件数 | 包装种类 | 毛重(公斤) | 净重(公斤)

集装箱号 | 随附单据 | 生产厂家

标记唛码及备注

项号 商品编号 商品名称、规格型号 数量及单位 最终目的国(地区) 单价 总价 币制 征免

(a)

(b)

Attach Form(s) W-2 here.

3	Wages, salaries, tips, etc. Attach Form(s) W-2 (see page 4)				3	15,460	26
4	Taxable refunds, credits, or offsets of state and local income taxes (see page 4)				4	0	00
5	Scholarship and fellowship grants. Attach explanation (see page 4)				5	0	00
6	Total wages and scholarships exempt by a treaty from page 2, Item J . .	6	5,000	00	▨		
7	Add lines 3, 4, and 5				7	15,460	26
8	Student loan interest deduction (see page 5)	8	0	00	▨		
9	Scholarship and fellowship grants excluded (see page 6)	9	0	00	▨		
10	**Adjusted gross income.** Subtract the sum of line 8 and line 9 from line 7				10	15,460	26
11	**Itemized deductions.** Enter state and local income taxes paid. Residents of India, see page 6				11	1,396	27
12	Subtract line 11 from line 10.				12	14,060	99
13	Exemption deduction (see page 6).				13	3,000	00
14	**Taxable income.** Subtract line 13 from line 12				14	11,060	99
15	**Tax.** Find your tax in the Tax Table on pages 11–15				15	1,361	00
16	Social security and Medicare tax on tip income not reported to employer. Attach Form 4137				16	0	00
17	Add lines 15 and 16. This is your **total tax** . . . . ▶				17	1,361	00
18	Federal income tax withheld (from Form W-2 and/or Form 1042-S) .	18	2,529	01	▨		
19	2002 estimated tax payments and amount applied from 2001 return	19	0	00	▨		
20	Credit for amount paid with Form 1040-C	20	0	00	▨		
21	Add lines 18 through 20. These are your **total payments** . . . . ▶				21	2,529	01

(c)

(d)

Figure 8: Some examples for model-based form frame line detection. (a) and (b) An export registration form used by the Customs Bureau of China and the corresponding line detection result. (c) and (d) A portion of a U.S. income tax return form and the corresponding line detection result. Red color shows the detected lines.

5.2 Rule Line Detection

In this section we use the method to detect severely broken rule lines. Different from the previous case, the number of lines is unknown, and the vertical line gaps may vary in different images due to the different style of the rule lined paper, or different scanning resolutions. However the length of lines and the vertical line gaps are roughly consistent in the same document image.

5.2.1 Vertical Line Gap Estimation

We need to estimate the average vertical line gap from the input image first. Since the line gaps between neighboring lines are roughly the same, the horizontal projection of rule lines is a periodic signal (the period is the average vertical line gap $\bar{g}$). We use an auto-correlation based approach to estimate the period of the projection. The auto-correlation of a signal x, with n samples $x(1), x(2), \cdots, x(n)$, is defined as

$$R(l) = \sum_{i=1}^{n-l} x(i)x(i+l) \quad l = 0, 1, \ldots, n-1 \tag{33}$$

The distance between the first two peaks of the auto-correlation is taken as the vertical line gap, as shown in Fig. 9.

To show the accuracy of the method, 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 the estimation errors of all vertical line gaps on a document page. For 168 images in our database, most page level estimation errors are within 0.5 pixels. The

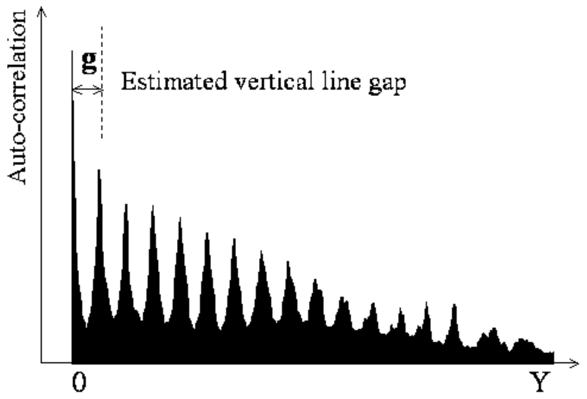


Figure 9: Vertical line gap estimation for rule line detection based on the auto-correlation of the projection profile.

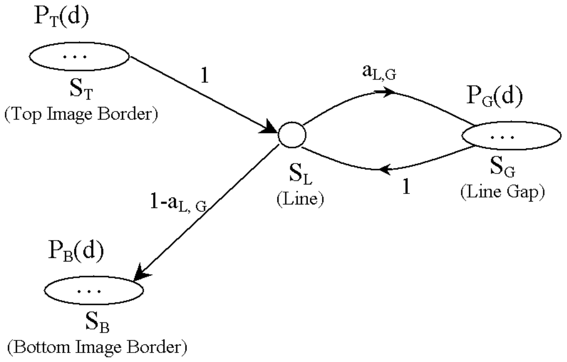


Figure 10: A simplified HMM model for rule line detection.

maximum page level estimation error is 1.3 pixels due to the large vertical line gap variance.

5.2.2 A Simplified Model

In order to reduce the complexity of the model (the number of states and parameters), we further simplify it by considering the special properties of rule lines. Since the vertical line gaps and the lengths of rule lines are roughly consistent in the same document image, we can merge states $S_{G,i}, i = 1, 2, \ldots, N-1$ into one state S_G, and $S_{L,i}, i = 1, 2, \ldots, N$ into another state S_L. The simplified model is shown in Fig. 10. State merging reduces the number of parameters significantly. Another advantage of such simplification is that we do not need to explicitly know how many lines are on a document.

5.2.3 Parameter Estimation

In our data set, the quality of different images are significantly different from each other. And the degradation of rule lines are different on the same image too. Therefore, we can not use the Gaussian distribution to model the projections of rule lines (the mixed Gaus-

Table 2: Observation probability distribution matrix B estimated from the training set containing 100 documents

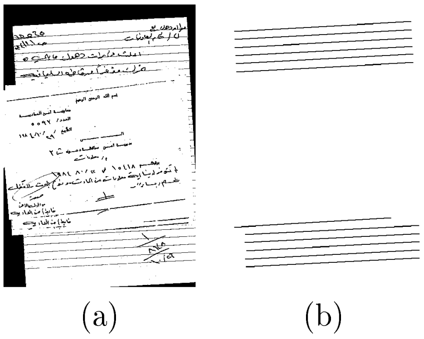
	0 Non-peak	1 $(0, \frac{w}{16}]$	2 $(\frac{w}{16}, \frac{w}{8}]$	3 $(\frac{w}{8}, \frac{w}{4}]$	4 $(\frac{w}{4}, w]$
S_L	106 (4.7%)	246 (10.8%)	378 (16.6%)	1,051 (46.2%)	493 (21.7%)
S_T, S_B, S_G	191,086 (98.8%)	2,052 (1.1%)	170 (0.1%)	58 (0.03%)	15 (0.008%)

sian distribution may be a good approximation). Instead, we quantize the observation into several levels, and estimate the probability of each quantized level directly from the groundtruthed data set. Peaks on the projection profile are of particular significance for line detection. Therefore, we first set all non-peaks on the profile to zero, then quantize the peaks on the projection profile into four levels using the following quantization levels: $w/16, w/8$, and $w/4$, where w is the image width. The observation probability distribution matrix B, estimated from the training set containing 100 documents, is listed in Table 2. Here, we let states S_T, S_B, and S_G, whose observations are the projection of text or noise, share the same observation distribution. We observed that 1) due to the severe brokenness, the horizontal projections of about 80% rule lines are less than 1/4 of the image width; 2) 4.7% rule lines do not form peaks; and 3) the peaks with small heights are more likely formed by text strokes or noise (2,052 instances), rather than by rule lines (246 instances). Therefore we need to use high level contextual information to achieve reasonable detection results for these severely broken lines.

We set duration probability of states S_T and S_B to the uniform distribution on $[0, \bar{g} - 1]$. The duration probability of state S_G is estimated directly from the groundtruth with the same method discussed in Section 5.1.2. With all these settings, the rule line detection accuracy on the training set is about 95.6%. For comparison, the accuracy is only 91.7% if we use the Gaussian distribution to approximate the observation. Since the parameters estimated from the training data are not optimal for the ultimate detection error criterion, the simplex method proposed by Nelder and Mead [25] is used to search the optimal parameter set which minimizes the detection error. Among parameters of our model, we only optimize the observation probability matrix B. Experiments show the detection accuracy increases to 97.3% on the training set after optimization.

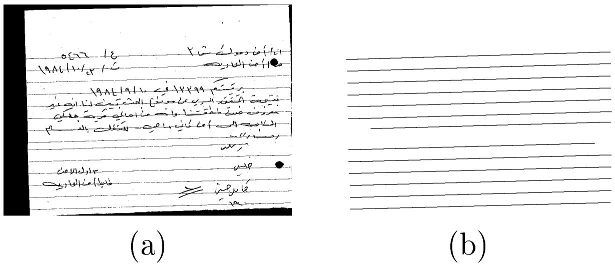
5.2.4 Examples

Fig. 1f shows the model-based line detection result for a rule lined document. Compared with Fig. 1e, we can see that with contextual information the result is significantly improved. Our model-based method is very robust even when the input images do not follow the model exactly. Fig. 11 shows an example: two pages are overlapped during scanning. Our algorithm still detects all rule lines correctly. In Fig. 12a, we remove 35 rows of the image (about half of the average vertical line gap of this document). The variation of the line gap is out of the range allowed by the HMM model. The corresponding detection result is shown in Fig. 12b, with only one line missed due to the anomalous vertical line gap.



(a) (b)

Figure 11: (a) The original image with two pages overlapped during scanning; (b) the corresponding line detection result.



(a) (b)

Figure 12: An example of a model mis-match. (a) A document image with 35 image rows removed; (b) the corresponding line detection result.

6 Experiments

In this section, we quantitatively evaluate the robustness of our line detection algorithm, and compare it with several non-model-based algorithms. We present our evaluation metrics in detail first.

6.1 Line Detection Evaluation Protocol

Line detection accuracy can be evaluated at the pixel level and the line level [23]. At pixel level we compare the difference of the pixels between groundtruthed and detected lines. It is straightforward and objective, but groundtruthing at the pixel level is extremely expensive when lines are broken, distorted and/or overlapped with text. Therefore, we evaluate the algorithm at the line level. Our evaluation metrics is based on the Hausdorff distance. The Hausdorff distance between two point sets is

$$H(A, B) = \max\{h(A, B), h(B, A)\} \tag{34}$$

where

$$h(A, B) = \max_{a \in A} \min_{b \in B} ||a - b|| \tag{35}$$

and $||.||$ is an underlying norm (e.g., the L_2 or Euclidean distance). The function $h(A, B)$ is called the *directed* Hausdorff distance from A to B. It identifies the point $a \in A$ that is the farthest from any point of B and measures the distance from a to its nearest neighbor

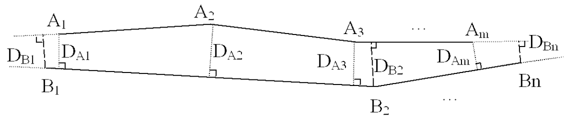


Figure 13: Hausdorff distance between two polylines.

in B [12]. The direct computation method for the Hausdorff distance is time consuming. However, for polyline representation, the Hausdorff distance can be easily calculated. Suppose polylines A and B are represented as a sequence of vertices $(A_1, A_2, \ldots, A_m)$ and $(B_1, B_2, \ldots, B_n)$ respectively, then the Hausdorff distance is simplified as

$$H(A, B) = \max\{H'(A, B), e(A, B)\} \tag{36}$$

where

$$\begin{aligned} H'(A, B) &= \max\{D_{A1}, D_{A2}, \ldots, D_{Am}, D_{B1}, D_{B2}, \ldots, D_{Bn}\} & (37) \\ e(A, B) &= \max\{||A_1 - B_1||, ||A_m - B_n||\} & (38) \end{aligned}$$

D_{Ai} is the perpendicular distance from A_i to polyline B, and D_{Bi} is the perpendicular distance from B_i to polyline A, as shown in Fig. 13. $H'(A, B)$ in Eq. (37) is the *perpendicular distance* between two polylines A and B, which evaluates the accuracy in determining the vertical location of a horizontal line and the horizontal location of a vertical line. $||A_i - B_j||$ is the Euclidean distance between points A_i and B_j. Suppose the vertices of a polyline are sorted from left to right for a horizontal line, and top to bottom for a vertical line. Then $||A_1 - B_1||$ and $||A_m - B_n||$ are the *end point distances.* Hausdorff distance $H(A, B)$ in Eq. (36) combines perpendicular distance and end point distance into one metric.

For severely broken lines, however, it is very hard to define the end points exactly. Therefore, we prefer to use two separate metrics: perpendicular distance and end point distance for evaluation, instead of a combined Hausdorff distance.

The absolute value of the end point distance is not suitable for evaluating both short and long lines. As a supplemental metric, the overlapping rate of polylines A and B

$$o(A, B) = \frac{\min\{A_m, B_n\} - \max\{A_1, B_1\}}{\max\{A_m, B_n\} - \min\{A_1, B_1\}} \tag{39}$$

is defined to evaluate the relative end point determination error.

For evaluation, we need to find the one-to-one correspondence between the detected and groundtruthed lines first. The method proposed in [20] is used to find such correspondence. Suppose the detected lines are $L_{d,i}, i = 1, 2, \ldots, m$, and the groundtruthed lines are $L_{g,j}, j = 1, 2, \ldots, n$. We want to find an optimal one-to-one mapping $F : \{L_{d,i}\} \rightarrow \{L_{g,j}\}$, such that

$$\sum_i dist(L_{d,i}, L_{g,F(i)}) \tag{40}$$

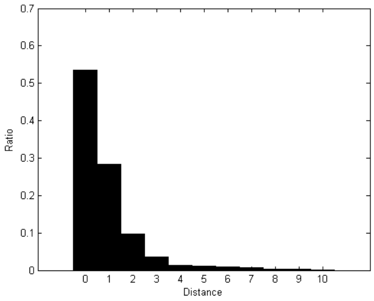


Figure 14: Histogram of the distances of groundtruth rule lines to the corresponding detected lines.

is minimized. The distance is the Hausdorff distance defined above. The optimal mapping can be achieved efficiently using the Hungarian algorithm [20]. For line detection, a correctly detected line should not be far away from its corresponding groundtruthed line. Let D_{max} be the threshold of the maximal tolerable localization error. The optimal matching problem in Eq. (40) is converted into a constrained optimization problem

$$dist(L_{d,i}, L_{g,F(i)}) \leq D_{max} \quad \text{for } i = 1, 2, \ldots, m \tag{41}$$

If the numbers of detected and groundtruthed lines are not equal, we add some dummy lines L_{dummy}, such that

$$\begin{aligned} dist(L_{d,i}, L_{dummy}) &= D_{max} \quad \text{for } i = 1, 2, \ldots, m \\ dist(L_{dummy}, L_{g,j}) &= D_{max} \quad \text{for } j = 1, 2, \ldots, n \end{aligned} \tag{42}$$

The missed groundtruthed lines and the falsely detected lines can be matched to the dummy lines. If the distance of a pair of matched lines is less than D_{min}, then the line is correctly detected; otherwise it is partially detected with a large location error. As suggested in [17], we set $D_{min} = 5$ and $D_{max} = 10$ pixels respectively.

6.2 Evaluation of Rule Line Detection

We obtained 168 Arabic document images with a total of 3,870 groundtruthed rule lines, most of which are severely broken. We use 100 images to train the HMM model, and the remaining 68 images as the test set. The detection results are shown in Table 3. On the test set, 96.8% lines are detected correctly and only 2 lines are missed. The histogram of the distances for all matched pairs is shown in Fig. 14. We can see that most of the detected lines (more than 90%) match the corresponding lines within 3 pixels. The false alarm rate is about 2.3%. Most of the false alarms are caused by the inconsistency between the detector and the subjective judgment of the groundtruther when lines are severely broken.

For correctly detected lines we evaluate the end point determination accuracy using the end point distance and overlapping rate defined in Eq. (38) and (39) respectively. The average end point distance is 6 pixels and the overlapping rate is 99.1%.

For comparison, we compared our model-based line detection algorithm with other non-model-based line detection algorithms: the Hough transform method, the projection

Table 3: Quantitative evaluation of the rule line detection result.

	Groundtruthed Lines	Detected Lines	Correct	Partial Correct	Missed	False Alarm
Training Set	2,274	2,319	2,212 (97.3%)	56 (2.5%)	6 (0.3%)	51 (2.2%)
Test Set	1,596	1,631	1,545 (96.8%)	49 (3.0%)	2 (0.1%)	37 (2.3%)

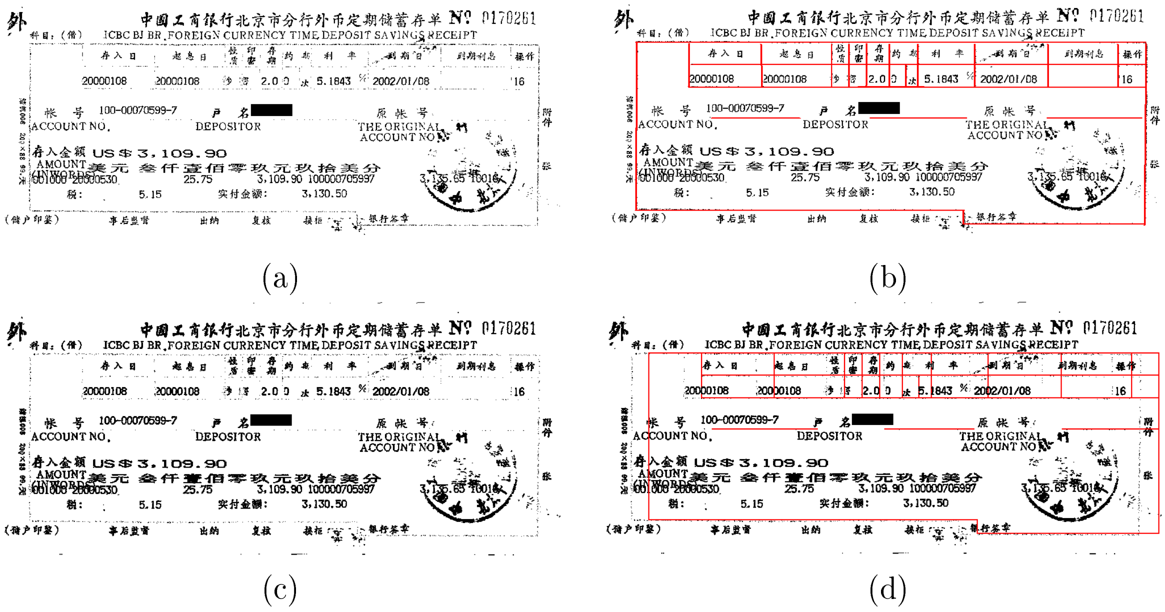
Table 4: Comparison of our model-based method with other methods on the test set (there are a total of 1,596 groundtruthed lines).

	Detected Lines	Correct	Partial Correct	Missed	False Alarm
Hough Transform	1,747	1,354 (84.8%)	113 (7.1%)	129 (8.1%)	280 (17.5%)
DSCC	2,162	1,398 (87.6%)	118 (7.4%)	80 (5.0%)	646 (40.5%)
Projection Method 1	1,969	1,384 (86.7%)	117 (7.3%)	95 (6.0%)	468 (29.3%)
Projection Method 2	4,387	1,472 (92.2%)	77 (4.8%)	47 (2.9%)	2,838 (177.8%)
Our Model-Based Method	1,631	**1,545 (96.8%)**	**49 (3.0%)**	**2 (0.1%)**	**37 (2.3%)**

based method, and the DSCC method. The line detection results on the test set with different algorithms are shown in Table 4. To reduce the false alarm rate of the Hough transform method, we restrict the search range of θ around 0^o after skew correction. For any pair of detected lines which are too close to each other, we remove the one with fewer black pixels. The accuracy of the Hough transform method is about 84.8%. Two variations of the projection method are implemented and compared. Projection methods 1 and 2 in Table 4 correspond to with and without close line removal respectively. We can see that without close line removal, the accuracy is as high as 92.2%, but the false alarm rate is also very high (177.8%). Close line removal significantly reduces the false alarm rate to 29.3%, but the accuracy drops to 86.7% as well. For the DSCC method, we restrict the merging direction to the horizontal direction. Since two DSCCs can be merged only when their heights are compatible [35], the false alarm rate of the DSCC method is much lower than the projection and Hough transform methods without close line removal. Therefore, we did not apply close line removal to the result of the DSCC method. As expected, our model based method achieved much better results in both accuracy and false alarm rate, due to the high level constraints between neighboring lines used. We observed that our model-based method can achieve higher accuracy than the projection method without close line removal, indicating that some severely broken rule lines are detected successfully even when they do not form peaks on the projection profile.

6.3 Evaluation of Known Form Processing

To evaluate the algorithm for known form processing, we scanned 100 bank deposit forms with reasonable image quality. We found in the experiment that only a few training samples are needed to achieve good result if the image quality is reasonable. We randomly selected 5 samples for training, and found that all the detected lines are within 8 pixels to the groundtruth without false alarms. Then we test the robustness of our method to image degradations. We randomly flip a certain ratio of black pixels on lines to white, and keep all pixels on text unchanged. Generally, the more severe the degradation is,



(a)

(b)

(c)

(d)

Figure 15: Degraded form documents and the corresponding line detection results. Red lines drawn on original images indicate the detected lines. (a) About 50% of the pixels of the lines are flipped; (b) line detection result of (a); (c) about 90% of pixels of the lines are flipped; (d) line detection result of (c).

the more accurate the model should be in order to detect lines correctly. Therefore, in the following experiments we increase the number of training samples. 50 forms are randomly selected for training and the remaining 50 forms for testing. Figs. 15a and 15c show the degraded images with 50% and 90% black pixels on lines flipped to white. As shown in Fig. 15b, the line detection result is perfect even half of the black pixels are flipped. Fig. 15d shows that the horizontal lines are still detected correctly even when 90% black pixels are flipped, but the vertical lines are misdetected. The detection accuracy vs. degradation on the test set is shown in Fig. 16. We can see that our method is very robust. It maintains good results with accuracy of 96.2% even when 80% black pixels of lines are flipped.

Our method is fast. The average processing time for an image with the size of

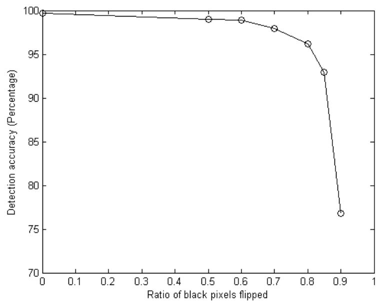


Figure 16: Line detection accuracy vs. ratio of black pixels flipped on lines.

1,700×1,800 pixels is about 0.4 seconds on a PC with 1.8GHZ CPU and 1GB memory. Most of the computation is spent on the preprocessing, such as skew angle estimation and correction.

7 Conclusion and Future Work

In this paper we present a novel approach to detect severely broken parallel lines in documents. Our method is based on a stochastic model to incorporate high level constraints into a general line detection algorithm. Instead of detecting lines individually, we use the Viterbi algorithm to detect all parallel lines simultaneously. Our method can detect 96.8% severely broken rule lines in the Arabic database we collected. Some challenging examples demonstrated the robustness of our approach. For known form processing, it can achieve good results even when about 80% black pixels on lines are flipped to white during degradation.

Our method can be extended for form identification easily. The probability of observing a sequence of observations given the model, which is useful for form identification, can be calculated with the forward algorithm efficiently. We are currently incorporating the method into a known form processing system for both form identification and registration. Our next focus is to develop a robust line removal algorithm to enhance the image quality of documents.

References

[1] 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.

[2] J.-L. Chen and H.-J. Lee. An efficient algorithm for form structure extraction using strip projection. *Pattern Recognition*, 31(9):1353–1368, 1998.

[3] D. Doermann. An introduction to vectorization and segmentation. In K. Tombre and A.K. Chhabra, editors, *Graphics Recognition – Algorithms and Systems, volume 1389 of Lecture Notes in Computer Science*, pages 1–8. Springer-Verlag, 1998.

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

[5] D. Dori and W. Liu. Sparse pixel vectorization: An algorithm and its performance evaluation. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 21(3):202–215, 1999.

[6] K.-C. Fan, J.-M. Lu, and G.-D. Chen. A feature point clustering approach to the recognition of form documents. *Pattern Recognition*, 31(9):1205–1220, 1998.

[7] G. Grimmett and D. Stirzaker. *Probability and Random Processes.* Oxford University Press, 2^{nd} edition, 2001.

[8] T.M. Ha and H. Bunke. Model-based analysis and understanding of check forms. *Int'l J. Pattern Recognition and Artificial Intelligence*, 8(5):1053–1081, 1994.

[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] O. Hori and S. Tanigawa. Raster-to-vector conversion by line fitting based on contours and skeletons. In *Proc. Int'l Conf. Document Analysis and Recognition*, pages 353–358, 1993.

[11] P.V.C. Hough. Machine analysis of bubble chamber pictures. In *Int'l Conf. High Energy Accelerators and Instrumentation*, 1959.

[12] D.P. Huttenlocher, G.A. Klanderman, and W.J. Rucklidge. Comparing images using the Hausdorff distance. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 15(9):850–863, 1993.

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

[14] J. Jimenez and J.L. Navalon. Some experiments in image vectorization. *IBM J. Res. Develop.*, 26:724–734, 1982.

[15] D. Jurafsky and J.H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition.* Prentice Hall, New Jersey, 2000.

[16] T. Kanungo, R.M. Haralick, and I.T. Phillips. Nonlinear local and global document degradation models. *Int'l J. Imaging Systems and Technology*, 5(4):220–230, 1994.

[17] B. Kong, I.T. Phillips, and R.M. Haralick. A benchmark: Performance evaluation of dashed-line detection algorithm. In *Proc. IAPR Int'l Workshop on Graphics Recognition*, pages 270–285, 1995.

[18] K.J. Lang, A.H. Waibel, and G.E. Hinton. A time-delay neural network architecture for isolated word recognition. *Neural Networks*, 3:23–43, 1990.

[19] S.E. Levinson. Continuously variable duration hidden Markov models for automatic speech recognition. *Computer, Speech and Language*, 1(1):29–45, 1986.

[20] G. Liu and R.M. Haralick. Optimal matching problem in detection and recognition performance evaluation. *Pattern Recognition*, 35(10):2125–2139, 2002.

[21] 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.

[22] J. Liu and A.K. Jain. Image-based form document retrieval. *Pattern Recognition*, 33(3):503–513, 2000.

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

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

[25] J. Nelder and R. Mead. A simplex method for function minimization. *Computer Journal*, 7:308–313, 1965.

[26] R. Plamondon and S.N. Srihari. On-line and off-line handwriting recognition: A comprehensive survey. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22(1):63–84, 2000.

[27] L.R. Rabiner. A tutorial on hidden Markov models and selected application in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[28] D.E. Rumelhart and J.L. McClelland. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition.* MIT Press, Cambridge, MA, 1986.

[29] H. Tamura. A comparison of line thinning algorithms from digital geometry viewpoint. In *Proc. Int'l Conf. Pattern Recognition*, pages 715–719, 1978.

[30] 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.

[31] A. Ting and M.K. Leung. Form recognition using linear structure. *Pattern Recognition*, 32(4):645–656, 1999.

[32] L.Y. Tseng and R.C. Chen. Recognition and data extraction of form documents based on three types of line segments. *Pattern Recognition*, 31(10):1525–1540, 1998.

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

[34] B. Yu, X. Lin, Y. Wu, and B. Yuan. Isothetic polygon representation for contours. *CVGIP*, 25:264–268, 1992.

[35] 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.