Preprocessing of Low-Quality Handwritten Documents Using Markov Random Fields

Huaigu Cao, *Member*, *IEEE*, and Venu Govindaraju, *Fellow*, *IEEE*

Abstract—This paper presents a statistical approach to the preprocessing of degraded handwritten forms including the steps of binarization and form line removal. The degraded image is modeled by a Markov Random Field (MRF) where the hidden-layer prior probability is learned from a training set of high-quality binarized images and the observation probability density is learned on-the-fly from the gray-level histogram of the input image. We have modified the MRF model to drop the preprinted ruling lines from the image. We use the patch-based topology of the MRF and Belief Propagation (BP) for efficiency in processing. To further improve the processing speed, we prune unlikely solutions from the search space while solving the MRF. Experimental results show higher accuracy on two data sets of degraded handwritten images than previously used methods.

Index Terms—Markov random field, image segmentation, document analysis, handwriting recognition.

1 Introduction

The goal of this paper is the preprocessing of degraded handwritten document images such as carbon forms for subsequent recognition and retrieval. Carbon form recognition is generally considered to be a very hard problem. This is largely due to the extremely low image quality. Although the background variation is not very intense, the handwriting is often occluded by extreme noise from two sources: 1) the extra carbon powder imprinted on the form because of accidental pressure and 2) the inconsistent force of writing. For example, people tend to write lightly at the turns of strokes. This is not a serious problem for writing on regular paper. However, when writing on carbon paper, the light writing causes notches along the stroke. Furthermore, most multipart carbon forms have a colored background so that the different copies can be distinguished. This results in very low contrast and a very low signal-to-noise ratio. Thus, the image quality of carbon copies is generally poorer than that of degraded documents that are not carbon copies. Therefore, binarizing the carbon copy images of handwritten documents is very challenging.

Traditional document image binarization algorithms [16], [15], [18], [11], [21] separate the foreground from the background by histogram thresholding and analysis of the connectivity of strokes. These algorithms, although effective, rely on heuristic rules of spatial constraints, which are not scalable across applications. Recent research [7], [8], [20] has applied the Markov random field (MRF) to document image binarization. Although these algorithms make various assumptions applicable only to low-resolution document images, we take advantage of the ability of the MRF to model spatial constraints in the case of high-resolution handwritten documents.

We present a method that uses a collection of standard patches to represent each patch of the binarized image from the test set. The input and output images are divided into nonoverlapping blocks (patches), and a Markov network is used to model the conditional dependence between neighboring patches. These representatives are obtained by clustering patches of binarized images in the training set. The use of representatives reduces the domain of the prior model to a manageable size. Since our objective is not image restoration (from linear or nonlinear degradation), we do not need an image/scene pair for learning the observation model. We can learn the observation model on the fly from the local histogram of the test image. Therefore, our algorithm achieves performance similar to adaptive thresholding algorithms [15], [18] even without using the prior model. As one might expect, the result improves with the inclusion of spatial constraints added by the prior model. In addition to binarization, we also apply our algorithm to the removal of form lines by modeling the way the probability density of the observation model is computed.

One significant improvement in this paper since our prior work [3] is the use of a more reliable method of estimating the observation model. This uses mathematical morphology to obtain the background, followed by Gaussian Mixture Modeling to estimate the foreground and background probability densities. Another improvement is the use of more efficient pruning methods to reduce the search space of the MRF effectively by identifying the patches that are surrounded by background patches. We present experimental results on the Prehospital Care Report (PCR) data set of handwritten carbon forms [14] and provide a quantitative comparison of word recognition rates on forms binarized by our method versus other approaches.

- *The authors are with the Center for Unified Biometrics and Sensors (CUBS), Department of Computer Science and Engineering, University at Buffalo, Amherst, NY 14260.*
E-mail: hcao@bbn.com, venu@cubs.buffalo.edu.

Manuscript received 26 June 2007; revised 13 Nov. 2007; accepted 6 May 2008; published online 14 May 2008.
Recommended for acceptance by D. Lopresti.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMI-2007-06-0389.
Digital Object Identifier no. 10.1109/TPAMI.2008.126.

2 Related Work

2.1 Locally Adaptive Methods for Binarization

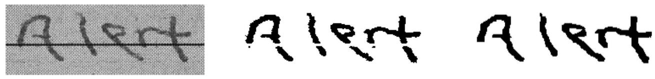
Usually, the quality of a document image is affected by variations in illumination and noise. By assuming that the background changes slowly, the problem of varying illumination is solved by adaptive binarization algorithms such as Niblack [15] and Sauvola [18]. The idea is to determine the threshold locally, using histogram analysis, statistical measures (mean, variance, etc.), or the intensity of the extracted background. Although noise can be reduced by smoothing, the resulting blurring affects handwriting recognition accuracy. Approaches of heuristic analysis of local connectivity, such as Kamel/Zhao [11], Yang/Yan [21], and Milewski/Govindaraju [14], solve the problem to some extent by searching for stroke locations and targeting only nonstroke areas. The Kamel/Zhao algorithm strokes by estimating the stroke width and then removes the noise in nonstroke areas using an interpolation and thresholding step. The Yang/Yan algorithm is a variant of the same method. The Milewski/Govindaraju algorithm examines neighboring blocks in orientations to search for nonstroke areas. However, in all of these approaches, the spatial constraints applied to the images are determined by a heuristic. Our objective is to find a probabilistic trainable approach to modeling the spatial constraints of the binarized image.

2.2 The Markov Random Field for Binarization

In recent years, inspired by the success of applying the MRF to image restoration [4], [5], [6], attempts have been made to apply MRF to the preprocessing of degraded document images [7], [8], [20]. The advantage of the MRF model over heuristic methods is that it allows us to describe the conditional dependence of neighboring pixels as the prior probability and to learn it from training data. Wolf and Doermann [20] defined the prior model on a 4×4 clique, which is appropriate for textual images in low-resolution video. However, for 300 dpi high-resolution handwritten document images, it is not computationally feasible to learn the potentials if we simply try to define a much larger neighborhood. Gupta et al. [7], [8] studied the restoration and binarization of blurred images of license plate digits. They adopted the factorized style of MRF using the product of compatibility functions [4], [5], [6], which are defined as mixtures of multivariate normal distributions computed over samples of the training set. They incorporated recognition into the MRF to reduce the number of samples involved in the calculation of the compatibility functions. However, this scheme also cannot be directly applied to unconstrained handwriting because of the larger number of classes and the low performance of existing handwriting recognition algorithms. In this paper, we describe an MRF adapted for handling handwritten documents that overcomes the computational challenges caused by high-resolution data and low accuracy rates of current handwriting recognizers.

2.3 Ruling Line Removal

The process of removing preprinted ruling lines while preserving the overlapping textual matter is referred to as



(a) (b) (c)

Fig. 1. Stroke-preserving line removal. (a) A word image with an underline across the text. (b) Binarized image with the underline removed. (c) Binarized image with the underline removed and strokes repaired.

image in-painting (Fig. 1) and is performed by inferring the removed overlapping portion of images from spatial constraints. MRF is ideally suited to this task and has been used successfully on natural scene images [2], [22]. Our task on document images is similar but more difficult: In both cases, spatial constraints are used to paint in the missing pixels, but the missing portions in document images often contain strokes with high-frequency components and details. Previously reported work on line removal in document images uses heuristic [1], [14], [23]. Bai and Huo [1] remove the underline in machine-printed documents by estimating its width. This works on machine-printed documents because the number of possible situations in which strokes and underlines intersect is limited. Milewski and Govindaraju [14] proposed restoring the strokes of handwritten forms using a simple interpolation of neighboring pixels. Yoo et al. [23] describe a sophisticated method that classifies the missing parts of strokes into different categories such as horizontal, vertical, and diagonal and connects them with runs (of black pixels) in the corresponding directions. It relies on many heuristic rules and is not accurate when strokes are lightly (tangentially) touching the ruling line.

3 Markov Random Field Model for Handwriting Images

We use an MRF model (Fig. 2) with the same topology as the one described in [5]. A binarized image x is divided into nonoverlapping square patches, $x_1, x_2, \ldots, x_N$, and the input image, or the observation y, is also divided into patches $y_1, y_2, \ldots, y_N$ so that x_i corresponds to y_i for any $1 \leq i \leq N$. Each binarized patch conditionally depends on its four neighboring binarized patches in both the horizontal and vertical directions, and each observed patch conditionally depends y on its corresponding binarized patch. Thus,

$$\begin{aligned}&\Pr(x_i|x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_N, y_1, \ldots y_N) = \\ &\quad \Pr(x_i|x_{n_1,i}, x_{n_2,i}, x_{n_3,i}, x_{n_4,i}), 1 \leq i \leq N,\end{aligned} \tag{1}$$

where $x_{n_1,i}$, $x_{n_2,i}$, $x_{n_3,i}$, and $x_{n_4,i}$ are the four neighboring vertices of x_i and

$$\begin{aligned}&\Pr(y_i|x_1, \ldots, x_N, y_1, \ldots, y_{i-1}, y_{i+1}, \ldots, y_N) = \\ &\quad \Pr(y_i|x_i), 1 \leq i \leq N.\end{aligned} \tag{2}$$

An edge in the graph represents the conditional dependence of two vertices. The advantage of such a patch-based topology is that relatively large areas of the local image are conditionally dependent. Our objective is to estimate the binarized

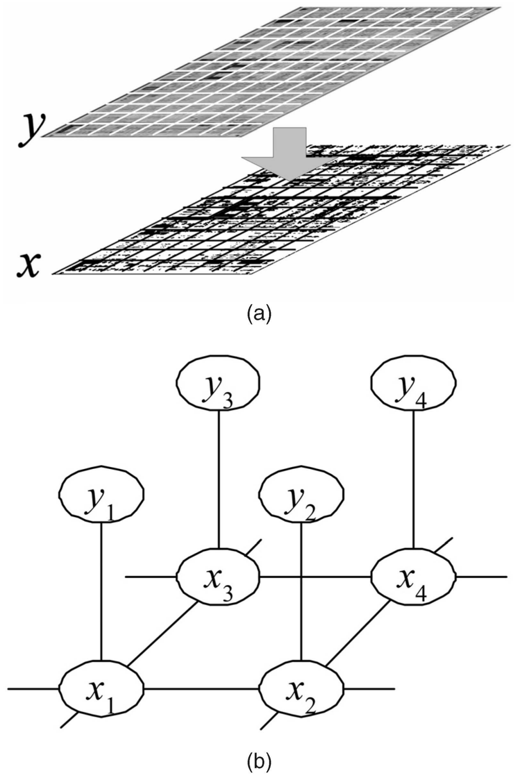


Fig. 2. The topology of the Markov network. (a) The input image y and the inferred image x. (b) The Markov network generalized from (a). In (b), each node x_i in the field is connected to its four neighbors. Each observation node y_i is connected to node x_i. An edge indicates the conditional dependence of two nodes.

image x from the posterior probability $\Pr(x|y) = \frac{\Pr(x,y)}{\Pr(y)}$. Since $\Pr(y)$ is a constant over x, we only need to estimate x from the joint probability $\Pr(x, y) = \Pr(x_1, \ldots, x_N, y_1, \ldots, y_N)$. This can be done by either the MMSE or the MAP approach [4], [5]. In the MMSE approach, the estimation of each x_j is obtained by computing the marginal probability:

$$\hat{x}_{jMMSE} = \sum_{x_j} x_j \times \sum_{x_1 \ldots x_{j-1} x_{j+1} \ldots x_N} \Pr(x_1, \ldots, x_N, y_1, \ldots, y_N). \tag{3}$$

In the MAP approach, the estimation of each x_j is obtained by taking the maximum of the probability $\Pr(x_1, \ldots, x_N, y_1, \ldots, y_N)$, i.e.,

$$\hat{x}_{jMAP} = \operatorname*{argmax}_{x_j} \max_{x_1 \ldots x_{j-1} x_{j+1} \ldots x_N} \Pr(x_1, \ldots, x_N, y_1, \ldots, y_N). \tag{4}$$

Estimation of the hidden vertices $\{x_j\}$ using (3) or (4) is referred to as inference. It is impossible to compute either (3) or (4) directly for large graphs because the computation grows exponentially as the number of vertices increases. We can use the Belief Propagation (BP) algorithm [17] to approximate the MMSE or MAP estimation in time linear in the number of vertices in the graph.

4 Inference in the MRF Using Belief Propagation

4.1 Belief Propagation

In the BP algorithm, the joint probability of the hidden image x and the observed image y from an MRF is represented by the following factorized form [5], [6]:

$$\Pr(x_1, \ldots, x_N, y_1, \ldots, y_N) = \prod_{(i,j)} \psi(x_i, x_j) \prod_k \phi(x_k, y_k), \tag{5}$$

where (i, j) are neighboring hidden nodes and ψ and ϕ are pairwise compatibility functions between neighboring nodes, learned from the training data. The MMSE and MAP objective functions can be rewritten as

$$\hat{x}_{jMMSE} = \sum_{x_j} x_j \times \sum_{x_1 \ldots x_{j-1} x_{j+1} \ldots x_N} \prod_{(i,j)} \psi(x_i, x_j) \prod_k \phi(x_k, y_k), \tag{6}$$

$$\hat{x}_{jMAP} = \operatorname*{argmax}_{x_j} \max_{x_1 \ldots x_{j-1} x_{j+1} \ldots x_N} \prod_{(i,j)} \psi(x_i, x_j) \prod_k \phi(x_k, y_k). \tag{7}$$

The BP algorithm provides an approximate estimation of $\hat{x}_{jMMSE}$ or $\hat{x}_{jMAP}$ in (6) and (7) by iterative steps. An iteration only involves local computation between the neighboring vertices. In the BP algorithm for MMSE, (6) is approximately computed by two iterative equations:

$$\hat{x}_{jMMSE} = \sum_{x_j} x_j \phi(x_j, y_j) \prod_k M_j^k, \tag{8}$$

$$M_j^k = \sum_{x_k} \psi(x_j, x_k) \phi(x_k, y_k) \prod_{l \neq j} \tilde{M}_k^l. \tag{9}$$

In (8), k runs over any of the four neighboring hidden vertices of x_j. M_j^k is the "message" passed from j to k and is calculated from (9) (the expression of M_j^k only involves the compatibility functions related to vertices j and k, so M_j^k can be thought of as the message passed from vertex j to vertex k). $\tilde{M}_k^l$ is M_k^l from the previous iteration. Note that M_j^k is actually a function of x_j. Initially, $M_j^k(x_j) = 1$ for any j and any value of x_j.

The formulas for the BP algorithm for MAP estimation are similar to (8) and (9) except that $\sum_{x_j} x_j$ and $\sum_{x_k}$ are replaced with $\operatorname{argmax}_{x_j}$ and $\max_{x_k}$, respectively:

$$\hat{x}_{jMAP} = \operatorname*{argmax}_{x_j} \phi(x_j, y_j) \prod_k M_j^k, \tag{10}$$

$$M_j^k = \max_{x_k} \psi(x_j, x_k) \phi(x_k, y_k) \prod_{l \neq j} \tilde{M}_k^l. \tag{11}$$

In our experiments, we use MAP estimation. The pairwise compatibility functions ψ and ϕ are usually heuristically defined as functions with the distance between two patches as the variable. We have found that a simple form is not suitable for binarized images because the distance can only take on a few values. Another way to select the form of ψ and ϕ is to use pairwise joint probabilities [4], [5]:

$$\psi(x_j, x_k) = \frac{\Pr(x_j, x_k)}{\Pr(x_j)\Pr(x_k)}, \tag{12}$$

$$\phi(x_k, y_k) = \Pr(x_k, y_k). \tag{13}$$

Replacing the ψ and ϕ functions in (10) and (11) with the definitions in (12) and (13), we obtain

$$\hat{x}_{j\,\mathrm{MAP}} = \operatorname*{argmax}_{x_j} \Pr(x_j)\Pr(y_j|x_j)\prod_k M_j^k \tag{14}$$

and

$$M_j^k = \max_{x_k} \Pr(x_k|x_j)\Pr(y_k|x_k)\prod_{l\neq j} \tilde{M}_k^l. \tag{15}$$

In order to avoid arithmetic overflow, we calculate the log values of the factors in (14) and (15):

$$L_j^k = \max_{x_k}\left(\log \Pr(x_k|x_j) + \log \Pr(y_k|x_k) + \sum_{l\neq j} \tilde{L}_k^l\right), \tag{16}$$

$$\hat{x}_{j\mathrm{MAP}} = \operatorname*{argmax}_{x_j}\left(\log \Pr(x_j) + \log \Pr(y_j|x_j) + \sum_k L_j^k\right), \tag{17}$$

where $L_j^k = \log M_j^k$, $\tilde{L}_k^l = \log \tilde{M}_k^l$, and the initial values of $\tilde{L}_j^k$s are set to zero.

To use (14) and (15), the probabilities $\Pr(x_j)$ and $\Pr(x_k|x_j)$ (prior model) and the observation probability density $\Pr(y_j|x_j)$ (observation model) have to be estimated.

4.2 Learning the Prior Model $\Pr(x_j)$ and $\Pr(x_k|x_j)$

The prior probabilities $\Pr(x_j)$ and $\Pr(x_k|x_j)$ are learned from a training set of clean handwriting images. The training set contains three high-quality binarized handwriting images from different writers. We can extract about two million patch images from these samples. Some samples from the training set are shown in Fig. 5. For training, we use clean samples because unlike the observed image, the hidden image should have good quality.

Assuming that the size of a patch is $B \times B$, the number of states of a binarized patch x_j is 2^{B^2}. If $B = 5$, for example, there will be about 34 million states. This makes searching for the maximum in (15) intractable. In order to solve this problem, we convert the original set of states to a much smaller set and then estimate the probabilities over the smaller set of states. Normally, this is done by dimension reduction using transforms like PCA. It is difficult, however, to apply such a transform to binarized images. Therefore, we use a number of standard patches to represent all of the 2^{B^2} states. This is similar to vector quantization (VQ) used in data compression. The set of representatives is referred to as the VQ codebook. Our method is inspired by the idea that images of similar objects can be represented by a very small number of the shared patches in the spatial domain.

Recently, Jojic et al. [10] explored this possibility of representing an image by shared patches. Similarly, the binarized document images with handwriting of nearly the same stroke width under the same resolution can also be decomposed into patches that appear frequently (Fig. 3). The representatives are learned by clustering all the patches

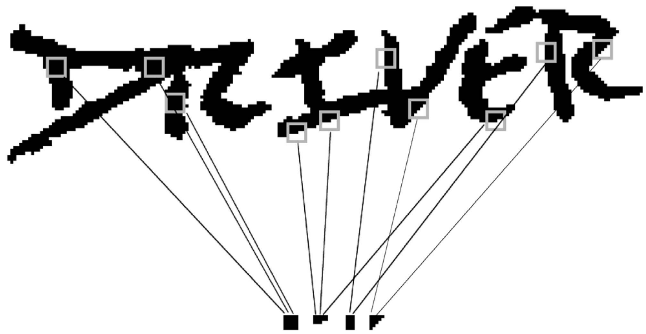


Fig. 3. Shared patches in a binary document image.

in our training set. We use the following approach: After every iteration of K-Means clustering, we round all the dimensions of each cluster center to zero or one. Given a training set of $B \times B$ binary patches, represented by $\{p_i\}$, we run the K-Means clustering starting with 1,024 clusters and remove the duplicate clusters and clusters containing less than 1,000 samples. The remaining cluster centers are taken as the representatives.

If the codebook is denoted by $\widetilde{C} = \{C_1, C_2, \ldots, C_M\}$, where $C_1, \ldots, C_M$ are M representatives, the error of VQ is given by the following equation:

$$\epsilon_{vq} = \frac{\sum_i \left[d(p_i, \widetilde{C})\right]^2}{\#\{p_i\} \cdot B^2}, \tag{18}$$

where $d(p_i, \widetilde{C})$ denotes the euclidean distance from p_i to its nearest neighbor(s) in $\widetilde{C}$ and $\#\{p_i\}$ denotes the number of elements in $\{p_i\}$. ϵ_{vq} is the square error normalized by the total number of pixels in the training set.

We can use the quantization error ϵ_{vq} to determine the parameter B. A larger patch size provides stronger local dependence, but it is difficult to represent very large patches because of the variety of writing styles exhibited by different writers. We tried different values of B ranging between five and eight, which coincide with the range of a typical stroke width in handwriting images scanned at 300 dpi, and chose the largest value of B that led to an ϵ_{vq} that is below 0.01. Thus, we determined the patch size $B = 5$. Then, the representation error $\epsilon_{vq} = 0.0079$ and 114 representatives are generated (Fig. 4). The size of the search space of a binarized patch is reduced from 2^{5^2} (about 34 million) to 114.

Now, we can estimate the prior probability $\Pr(x_j)$ over codebook $\widetilde{C}$,

$$\sum_{l=1}^{M} \Pr(x_j = C_l) = 1 \tag{19}$$

so that the prior probabilities $\Pr(x_j)$ over the reduced search space must add up to one. We estimate $\Pr(x_j)$ from the relative size of the cluster centered at C_l. A patch p_i from the training set is a member of cluster C_l $(1 \leq l \leq M)$ if C_l is a nearest neighbor of p_i among all of $C_1, \ldots, C_M$ and is denoted by $p_i \in C_l$. Note that a patch p_i from the training set may have multiple nearest neighbors among $C_1, \ldots, C_M$. The number of nearest neighbors of p_i in $\widetilde{C}$ is denoted by $n_{\widetilde{C}}(p_i)$. Thus, the probability $\Pr(x_j)$ is estimated by

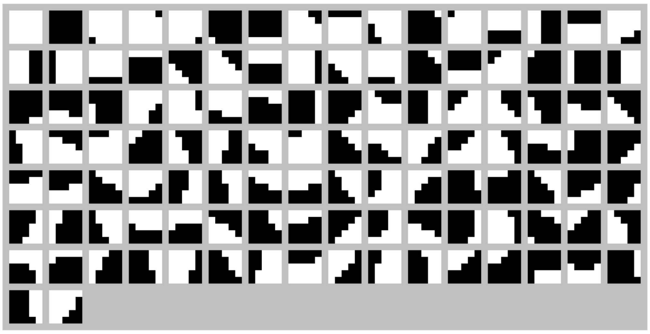


Fig. 4. The 114 representatives of shared patches obtained from clustering.

$$\hat{\Pr}(x_j = C_l) = \frac{\sum_{p_i \in C_l} \frac{1}{n_{\widetilde{C}}(p_i)}}{\#\{p_i\}}, \quad l = 1, 2, \ldots, M, \tag{20}$$

where $\#\{p_i\}$ is the number of patches in $\{p_i\}$. $\hat{\Pr}(x_j = C_l)$ in (20) is estimated by the size of cluster C_l normalized by the total number of training patches. It is easy to verify that the probabilities in (20) add up to one.

$\Pr(x_j, x_k)$ are estimated in the horizontal and vertical directions, respectively. Similar to (20), the $\Pr(x_j, x_k)$ $(x_j, x_k \in \widetilde{C})$ in horizontal direction is estimated by

$$\hat{\Pr}(x_j = C_{l_1}, x_k = C_{l_2}) = \frac{\sum_{\substack{(p_{i_1}, p_{i_2}),\\ p_{i_1} \in C_{l_1}, p_{i_2} \in C_{l_2}}} \frac{1}{n_{\widetilde{C}}(p_{i_1}) n_{\widetilde{C}}(p_{i_2})}}{\#\{(p_{i_1}, p_{i_2})\}}, \tag{21}$$
$$l_1 = 1, 2, \ldots, M, \; l_2 = 1, 2, \ldots, M,$$

where (p_{i_1}, p_{i_2}) runs for all pairs of patches in the training set $\{p_i\}$ such that p_{i_1} is the left neighbor of p_{i_2} and $\#\{(p_{i_1}, p_{i_2})\}$ is the number of pairs of left-and-right neighboring patches in $\{p_i\}$.

The $\Pr(x_j, x_k)$ $(x_j, x_k \in \widetilde{C})$ in the vertical direction is estimated by an equation similar to (21) except that p_{i1} is the upper neighbor of p_{i2}.

4.3 Learning the Observation Model $\Pr(y_j|x_j)$

The observation model on the pixel level can be estimated from the distribution of gray-scale densities of pixels [20]. For the patch-level observation model, we need to map the single-pixel version to the vector space of patches. The pixels of an observed patch y_j are denoted by $y_j^{r,s}$, $1 \le r, s \le 5$. The pixels of a binarized patch x_j are denoted by $x_j^{r,s}$, $1 \le r, s \le 5$. We assume that the pixels inside an observed patch y_j and the respective binarized patch x_j obey a similar conditional dependence assumption as the patches in the patch-based topology (2), i.e.,

$$\begin{aligned} &\Pr\left(y_j^{r,s} | y_j^{1,1}, \ldots, y_j^{r,s-1}, y_j^{r,s+1}, \ldots, y_j^{5,5}, x_j^{1,1}, \ldots, x_j^{5,5}\right) \\ &= \Pr\left(y_j^{r,s} | x_j^{r,s}\right), \quad 1 \le r, s \le 5. \end{aligned} \tag{22}$$

Thus, it can be proven that

$$\Pr\left(y_j^{1,1}, \ldots, y_j^{5,5} | x_j^{1,1}, \ldots x_j^{5,5}\right) = \prod_{r=1}^{5} \prod_{s=1}^{5} \Pr\left(y_j^{r,s} | x_j^{r,s}\right). \tag{23}$$

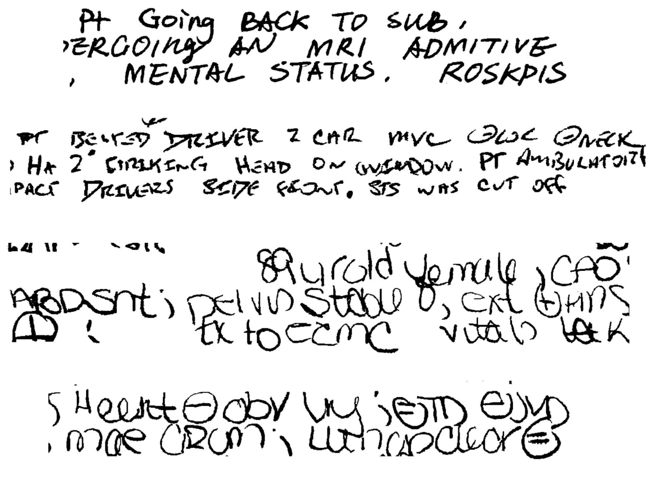


Fig. 5. Binarized images from three writers for learning the prior model.

Given the distribution of the intensity of foreground (strokes) $p_f(y_j^{r,s}) = \Pr(y_j^{r,s} | x_j^{r,s} = 0)$ and the distribution of the intensity of background $p_b(y_j^{r,s}) = \Pr(y_j^{r,s} | x_j^{r,s} = 1)$, according to (23), the conditional p.d.f $\Pr(y_j|x_j)$ is calculated as

$$\Pr(y_j|x_j) = \prod_{1 \le r,s \le 5, x_j^{r,s}=0} p_f\left(y_j^{r,s}\right) \prod_{1 \le r,s \le 5, x_j^{r,s}=1} p_b\left(y_j^{r,s}\right). \tag{24}$$

The expression $1 \le r, s \le 5$, $x_j^{r,s} = 0$ means that the scope of the product is for any r and s such that $1 \le r, s \le 5$ and $x_j^{r,s} = 0$. The expression $1 \le r$, $s \le 5$, $x_j^{r,s} = 1$ is specified in the same way.

The probability densities p_f and p_b change over an image while the intensity of the background is changing. However, this is not a problem as we can use regularization techniques such as Background Surface Thresholding (BST) [19] to obtain the background and normalize the images. This background mapping technique is equivalent to adaptive thresholding algorithms such as the Niblack algorithm [15].

Learning the probability density functions p_f and p_b is unsupervised. Assuming that p_f and p_b are two normal distributions, one way to compute p_f and p_b is given as follows: First, we determine a threshold T by an adaptive thresholding method such as the Niblack algorithm. Then, we use all of the pixels with gray level $\le T$ to estimate the mean and variance of p_f and use the remaining pixels to estimate the mean and variance of p_b. This method for estimating the observation probability densities is affected by the sharp truncation of "tails" in both normal distributions. Instead, we estimate the densities by modeling them as a two-Gaussian Mixture Model (2-GMM) using the Expectation-Maximization (EM) algorithm. The 2-GMM is not always reliable, due to the fact that the signals are not strictly Gaussian and that the algorithm is unsupervised with respect to the foreground/background categories. Our strategy is to get a reliable estimation of the p.d.f. of the background by background extraction and refine it when fitting the mixture model. Our algorithm is described as follows:

1. *Background extraction.* Estimate the mean μ and variance σ^2 of the entire input image. Binarize the

image using threshold $thr = \mu - 2\sigma$ and dilate the foreground with a 4×4 template. We mark the background pixels in the original image using the binarized image and estimate the mean μ_{b_0} and variance σ_{b_0} of density p_b from the extracted background pixels.

2. *EM algorithm for estimating the 2-GMM.* Suppose K gray-scale pixel samples from the image $z_1, z_2, \dots, z_K$ are available and their distribution is $\Lambda Z_1 + (1 - \Lambda) Z_2$, where Z_1, Z_2, and Λ are three random variables, $Z_1 \sim N(\mu_f, \sigma_f^2)$, $Z_2 \sim N(\mu_b, \sigma_b^2)$, $\Lambda \in \{0, 1\}$, and $\Pr(\Lambda = 1) = \lambda$. Denote the density of a normal distribution $N(\mu, \sigma^2)$ by $n_{\mu,\sigma^2}(y)$.

   Initial values: $\hat{\mu}_f = \mu_{b_0}/2$, $\hat{\mu}_b = \mu_{b_0}$, $\hat{\sigma}_f = \hat{\sigma}_b = 10.0$, and $\hat{\lambda} = 0.5$.

   E-step: Obtain the expectation of λ for every sample:

$$\hat{\lambda}_i = \frac{\hat{\lambda} \cdot n_{\hat{\mu}_f, \hat{\sigma}_f^2}(z_i)}{\hat{\lambda} \cdot n_{\hat{\mu}_f, \hat{\sigma}_f^2}(z_i) + (1 - \hat{\lambda}) \cdot n_{\hat{\mu}_b, \hat{\sigma}_b^2}(z_i)}, \quad i = 1, 2, \dots, K. \tag{25}$$

   M-step: Update the foreground mean and variance:

$$\hat{\mu}_f = \frac{\sum_{i=1}^{K} \hat{\lambda}_i \cdot z_i}{\sum_{i=1}^{K} \hat{\lambda}_i}, \hat{\sigma}_f^2 = \frac{\sum_{i=1}^{K} \hat{\lambda}_i \cdot (z_i - \hat{\mu}_f)^2}{\sum_{i=1}^{K} \hat{\lambda}_i}, \tag{26}$$

   and the prior

$$\hat{\lambda} = \sum_{i=1}^{K} \hat{\lambda}_i / K. \tag{27}$$

   Repeat the above E-step and M-step until the algorithm converges.

The comparison of the two methods for p.d.f. estimation is shown in Fig. 6. The p.d.f. estimation algorithm using the EM algorithm has an advantage over the algorithms using Niblack thresholding because it avoids the problem of sharply cutting the histogram and has a smoother estimation at the intersection of two Gaussian distributions.

Note that we assume that the image is bimodal. Our work focuses on document images where the bimodal assumption generally holds. If it does not hold, it may be possible to perform a color-image segmentation of the page and use a local histogram to binarize the image.

4.4 Ruling Line Removal

First, the ruling lines are located by template matching; this is relatively straightforward to implement because of the fixed form layout and is true for most types of forms in other applications as well. Therefore, we can define a Boolean mask m such that

$$m(j, r, s) = true \Longleftrightarrow \text{pixel } y_j^{r,s} \text{ is within any of the ruling lines.} \tag{28}$$

We only need to make a minor modification to (24) for the ruling line removal:

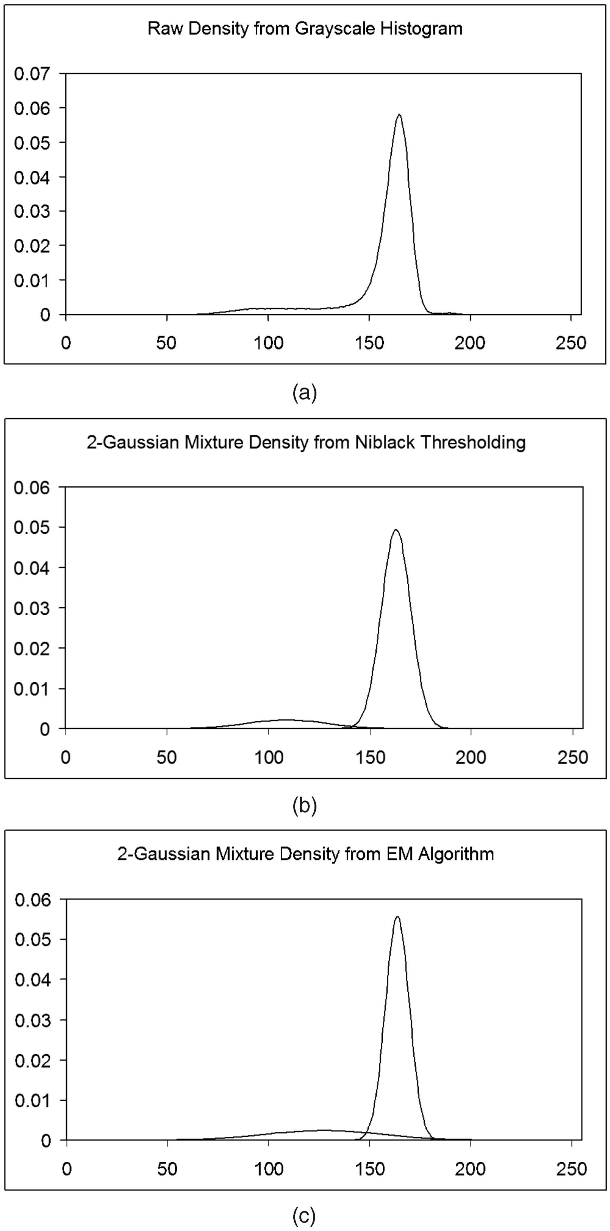


Fig. 6. The smoothed gray-scale histogram and estimated foreground and background p.d.f. using two methods. The method using Niblack thresholding did not perform well at the intersection of the two density functions, whereas the method using the EM algorithm improved the result. (a) Raw density by smoothing the gray-scale histogram of the image in Fig. 7. (b) Two-Gaussian mixture density from thresholding. (c) Two-Gaussian mixture density from the EM algorithm.

$$\Pr(y_j|x_j) = 1 \times \prod_{\substack{1 \le r,s \le 5, x_j^{r,s}=0, \\ \text{and } m(j,r,s)=false}} p_f\left(y_j^{s,t}\right) \times \prod_{\substack{1 \le r,s \le 5, x_j^{r,s}=1, \\ \text{and } m(j,r,s)=false}} p_b\left(y_j^{s,t}\right). \tag{29}$$

The probability $\Pr(y_j|x_j)$ in (29) is one if $m(j, r, s)$ is always $true$ for any r and s in the jth patch. We replace (24) with (29) for the compound tasks of binarization and line removal.

4.5 Pruning the Search Space of MRF Inference

To this point, MRF-based preprocessing has been presented as a self-contained general-purpose algorithm. To make this approach tractable, we adopt a patch-based strategy and reduce the search space using VQ. Initially, the size of the search space of every patch x_i in (4) and (5) is 2^{25}. Thus, the domain of every variable x_i is

$\{\underbrace{00\ldots0}_{25\ 0s}, \underbrace{00\ldots0}_{24\ 0s}1, \ldots, \underbrace{11\ldots1}_{25\ 1s}\}$, and we reduce the search space to $\widetilde{C} = \{C_1, C_2, \ldots, C_{114}\}$ by VQ. Although the amount of computation is reduced by the above strategies, the MRF algorithm is still slower than traditional binarization algorithms. There are ways to make the algorithm faster. Next, we will describe a technique to prune the search space of each x_i. After pruning, a number of elements are removed from $C_1, C_2, \ldots, C_{114}$ to make an even smaller search space of x_i.

The number of possible values per patch (114) can be reduced by pruning the smaller posterior probabilities $\Pr(x_j = C_l|y)$ calculated using (14) after each iteration, i.e.,

$$\Pr(x_j = C_l|y) = \frac{\Pr(x_j = C_l)\Pr(y_j|x_j = C_l)\prod_k M_j^k(C_l)}{\sum_{m=1}^{114}\left(\Pr(x_j = C_m)\Pr(y_j|x_j = C_m)\prod_k M_j^k(C_m)\right)} \quad (l = 1, 2, \ldots, 114), \tag{30}$$

where $M_j^k(C_l)$ is the message from x_j to x_k when $x_j = C_l$. However, this pruning is not safe on patches containing paint-in pixel(s). Due to the lack of observations of these pixels, it will take several iterations for them to converge to the right values, which may have very small posterior probabilities in the first one or two iterations. Therefore, the right values tend to be pruned incorrectly if we prune aggressively. In order to reduce the number of states of the patches containing paint-in pixel(s), we use a heuristic method to identify the patches surrounded by background and prune their search space. This method is effective due to the higher prior probability of the background (white patches).

From the above analysis, we have the following two-step strategy to accelerate the algorithm:

1. Find a global threshold thr_{prune} such that 90 percent of the pixels in the test image are below thr_{prune}. This is done by solving

$$\frac{\hat{\lambda}\cdot n_{\hat{\mu}_f,\hat{\sigma}_f^2}(thr_{prune})}{\hat{\lambda}\cdot n_{\hat{\mu}_f,\hat{\sigma}_f^2}(thr_{prune}) + (1-\hat{\lambda})\cdot n_{\hat{\mu}_b,\hat{\sigma}_b^2}(thr_{prune})} = 90\%. \tag{31}$$

For any patch $x_j\ (1 \le j \le N)$ of the binarized image, define a pruning mask $PRUNE_j(l)$, $(l = 0, 1, \ldots, 114)$. If $PRUNE_j(l)$ is $true$, C_l is pruned from the search space for solving x_j. Given a patch x_j and observed patch y_j centered at j_0, the pruning mask of x_j is initialized as $PRUNE_j(1) = false, PRUNE_j(2) = \ldots = PRUNE_j(114) = true$ if every observed pixel within a 9 × 9 neighborhood of j_0 is either above thr_{prune} or is marked for in-painting. Thus, all possible values of x_j will be pruned except the pure white patch. Otherwise, the pruning mask is initialized as $PRUNE_j(1) = PRUNE_j(2) = \ldots = PRUNE_j(114) = false$.

2. In each iteration, skip any C_l in the search spaces of x_j or x_k in (14) and (15) if $PRUNE_j(l)$ or $PRUNE_k(l)$ is $true$. Thus, (14) becomes

$$\hat{x}_{j\,\mathrm{MAP}} = \operatorname*{arg\,max}_{\substack{x_j,\\ PRUNE_j(x_j)\text{ is }false}} \Pr(x_j)\Pr(y_j|x_j)\prod_k M_j^k. \tag{32}$$

Equation (15) becomes

$$M_j^k(x_j) = \max_{\substack{x_k,\\ PRUNE_k(x_k)\text{ is }false}} \Pr(x_k|x_j)\Pr(y_k|x_k)\prod_{l\neq j}\tilde{M}_k^l, \quad \text{if } PRUNE_j(x_j)\ is\ false. \tag{33}$$

After each iteration, update the posterior probabilities $\Pr(x_j = C_l|y)$:

$$\Pr(C_l|y) = \frac{L_l}{\sum_l L_l} \quad (l = 1, 2, \ldots, 114), \tag{34}$$

where

$$L_l = \begin{cases} \Pr(x_j = C_l)\Pr(y_j|x_j = C_l)\prod_k M_j^k(C_l), \\ \quad \text{if } PRUNE_j(l) \text{ is } true, \\ 0, \text{ otherwise.} \end{cases} \tag{35}$$

Switch any $PRUNE_j(l)\ (l = 1, 2, \ldots, 114)$ to $true$ if $\Pr(x_j = C_l|y) < \Pr_{min}$, where $\Pr_{min}$ is the pruning threshold. Larger values of $\Pr_{min}$ make the algorithm faster and less accurate.

We will show experimentally how a different $\Pr_{min}$ affects the accuracy and the speed of the proposed algorithm. In general, we should choose a small $\Pr_{min}$ so that the algorithm does not prune excessively. For the patches that contain pixels to in-paint, $\Pr_{min}$ should be greater than the prior probability of any state in the codebook, i.e., $\Pr_{min} < \min_l \Pr(C_l)$, so that any state will not be pruned in the first iteration of BP.

5 Experimental Results and Analysis

5.1 Test Data Sets

Our test data includes the PCR carbon forms and handwriting images from IAM database 3.0 [13]:

1. *PCR forms.* In New York state, all patients who enter the Emergency Medical System (EMS) are tracked through their prehospital care to the emergency room using the PCRs. The PCR is used to gather vital patient information. The PCR forms are scanned as color images at 300 dpi. Handwriting recognition on this data set is quite challenging for several reasons:

   a. Handwritten responses are very loosely constrained in terms of writing style due to urgent emergency situations.
   b. Images are scanned from noisy carbon copies, and color background leads to low contrast and a low signal-to-noise ratio (Fig. 8).
   c. The (preprinted) ruling lines often intersect text.

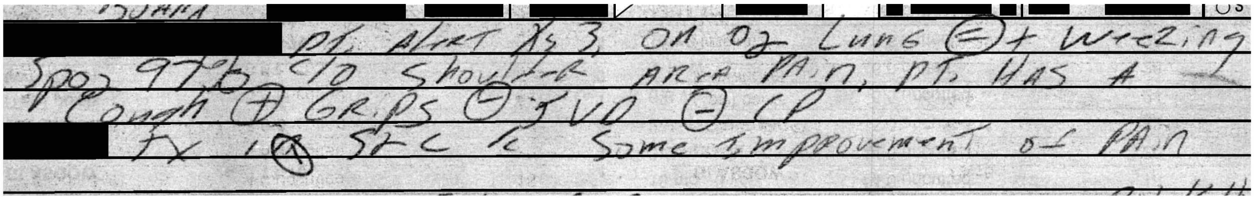


Fig. 7. A sample patch cropped from a carbon image in our test set. All pixels we intend to paint in are marked in black.

   d. Medical lexicons of words are large (more than 4,000 entries).

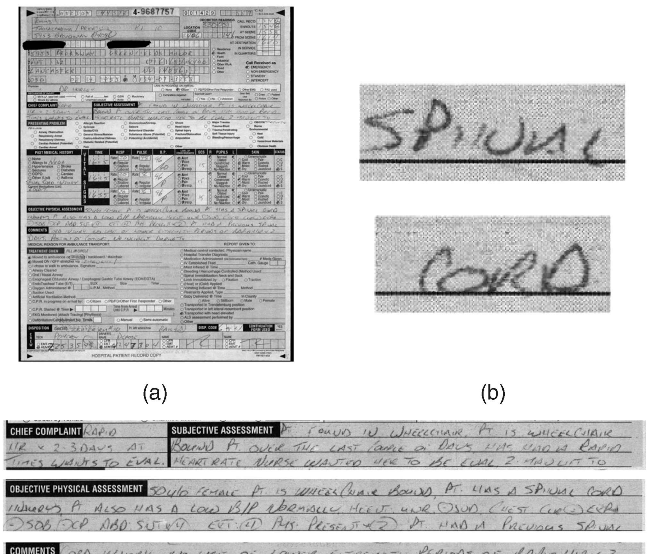
   Very low word recognition rates (below 20 percent) have been reported on this data set [14].An example of the handwritten text and pre-printed ruling lines in the PCR forms is shown in Fig. 8.

2. *IAM database.* The IAM database contains high-quality images of unconstrained handwritten English text, which were scanned as gray-scale images at 300 dpi. Using rough estimates, the content of the database can be summarized as follows:

   a. 500 writers who contributed samples of their handwriting,
   b. 1,500 pages of scanned text,
   c. 10,000 isolated and labeled text lines,
   d. 100,000 isolated and labeled words.

5.2 Display of Preprocessing Results

First, we applied our algorithm to the input image shown in Fig. 7. This input image is cropped from a PCR form. By aligning the input image with a template form image, rough estimations of the positions of lines and unwanted machine-printed blocks are detected. Then, we move the lines and machine-printed regions locally to find the best matches of unwanted pixels and marked these pixels in black. Our test images and the images for training the prior model are from different writers. It is clear that the writing style in Fig. 7 is not like any of the styles in Fig. 5. The results after iterations 1, 2, 4, and 16 of BP ran on Fig. 7 are shown in Fig. 9. After the first iteration, the message has not yet been passed between neighbors. The edges of strokes are jagged due to noisy background and errors in the VQ discussed in Section 4.2. All of the preprinted lines are dropped. After two iterations, text edges are smoothed, but most lines are not fully restored. After four iterations, nearly all of the strokes are restored, although a few tiny artifacts are still visible. After 16 iterations, the artifacts are mostly removed.



(a) (b)

(c)

Fig. 8. An example of PCR forms. (a) An entire PCR form. (b) A small local region showing obscured text and background noise array. (c) Fields of interest in the PCR form.

5.3 Results of Acceleration: Speed versus Accuracy

We have tested the effect of different values of parameter Pr_{min} on the speed and accuracy of our algorithm using the PCR carbon form image in Fig. 7 and the IAM handwriting image in Fig. 10. In order to compare the results obtained by our algorithm with different values of Pr_{min}, we have taken the output images of $\mathrm{Pr}_{min} = 0$ (which indicates no speedup) as reference images and have counted the pixels in the output images with various Pr_{min}s that are different from the reference images. The results are shown in Table 1. The runtimes are obtained on a PC with an Intel 2.8 GHz CPU.

In Table 1, even with a very small Pr_{min}, e.g., 10^{-8}, the runtime decreased significantly. The error rate of the low-quality PCR image is below 0.01 percent when $\mathrm{Pr}_{min} \leq 1 \times 10^{-6}$ and is zero when $\mathrm{Pr}_{min} \leq 1 \times 10^{-7}$. In Table 2, the error rate of the high-quality IAM image is below 0.01 percent when $\mathrm{Pr}_{min} \leq 1 \times 10^{-4}$ and is zero when $\mathrm{Pr}_{min} \leq 1 \times 10^{-5}$. In the following experiments where we compare OCR results, we chose $\mathrm{Pr}_{min} = 10^{-7}$.

5.4 Comparison to Other Preprocessing Methods

In Fig. 11, we compare our approach with the preprocessing algorithm of Milewski and Govindaraju [14], the Niblack algorithm [15], and the Otsu algorithm [16]. The Milewski/Govindaraju algorithm performs both binarization and line removal. The Niblack and Otsu algorithms are for binarization only. The text of the images shown in Fig. 11 is "67 yo ♀ pt found mfg X Ray." From the result of the MRF-based algorithm, the text "67 yo ♀ pt found" is clear and the text "MFG X ray" is obscured but some letters are still legible. In the output of the Milewski/Govindaraju algorithm, the words "pt," "mfg," "X," and "Ray" are not legible. The output of the Niblack algorithm is noisier although it retains some details of the foreground. The result of the Otsu algorithm is also very noisy and loses more foreground details than the Niblack algorithm. Fig. 12 shows that our line removal (Fig. 12c) achieves a smoother restoration of

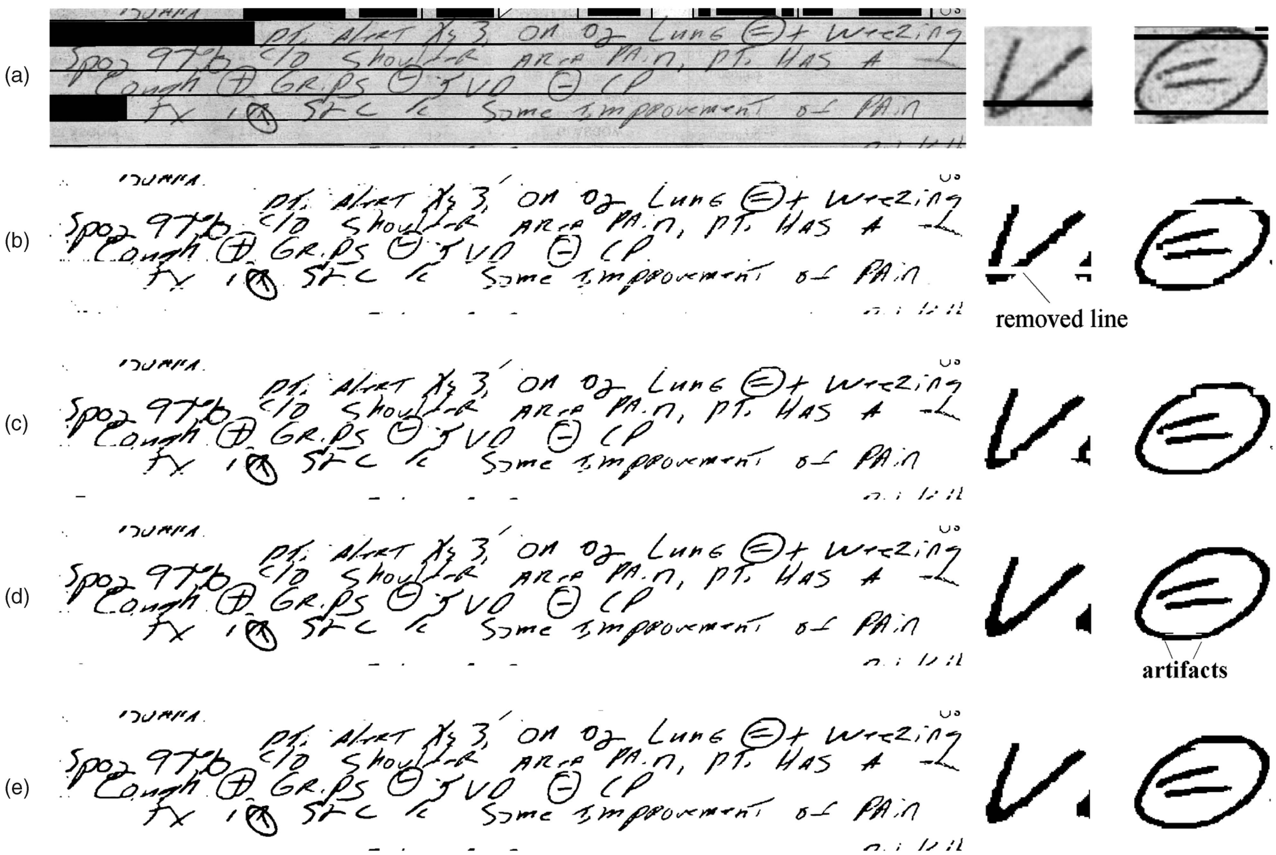


Fig. 9. The binarization and line removal result of the sample shown in Fig. 7. (a) Original image with lines marked in black. (b) Output after one iteration of BP. (c) Output after two iterations of BP. (d) Output after four iterations of BP. (e) Output after 16 iterations of BP.

strokes that touch the ruling line than the Milewski/Govindaraju algorithm (Fig. 12b).

In addition to the above qualitative comparison, we have also used handwriting recognition results to obtain a quantitative comparison. First, we tested the four algorithms on 100 PCR forms. All of the 3,149 binarized word images extracted from the 100 form images were recognized using the word recognition algorithm in [12] with a lexicon of 4,580 English words. We split the 3,149 word images into two sets: Set #1 contains 1,203 word images that are not affected by overlapping form lines, i.e., no intersection of stroke and line; Set #2 contains 1,946 pairs that are affected by form lines. Thus, the word recognition accuracy on Set #1 measures the performance of binarization only and can be used to compare all four algorithms.

We calculated the top-n ($n \geq 1$) recognition rates instead of only the top-one rate for comparison because top-n rates are of greater importance to the problem of indexing text that suffers from a very high error rate [9]. Table 3 shows that the MRF-based method results in higher overall recognition rates and also performs more efficient line removal.

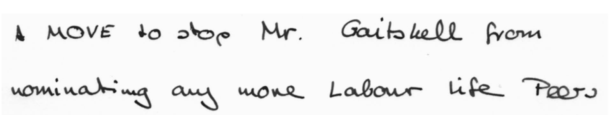


Fig. 10. A sample from the IAM database.

TABLE 1
Comparison of the Speed and Accuracy of the Proposed Algorithm over Different Values of Pr_{min} Tested on the PCR Carbon Form Image (2,420 $\times$ 370) in Fig. 7

Pr_{min}	Number of Different Pixels	Percentage of Different Pixels (%)	Time (sec)
0	0	0	3249
1×10^{-8}	0	0	204
1×10^{-7}	0	0	138
1×10^{-6}	56	0.0063	96
1×10^{-5}	145	0.016	72
1×10^{-4}	308	0.034	57
1×10^{-3}	1122	0.13	37

TABLE 2
Comparison of the Speed and Accuracy of the Proposed Algorithm over Different Values of Pr_{min} Tested on the IAM Image (2,124 $\times$ 369) in Fig. 10

Pr_{min}	Number of Different Pixels	Percentage of Different Pixels (%)	Time (sec)
0	0	0	1694
1×10^{-8}	0	0	29
1×10^{-7}	0	0	25
1×10^{-6}	0	0	24
1×10^{-5}	0	0	24
1×10^{-4}	14	0.002	23
1×10^{-3}	107	0.014	23

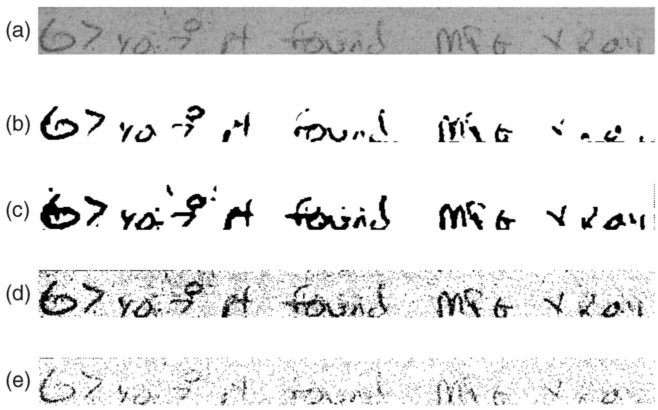


Fig. 11. Comparison of binarization results of the MRF-based algorithm versus three other algorithms. (a) Input image. (b) Output of the Milewski/Govindaraju algorithm. (c) Output of the MRF-based algorithm. (d) Output of the Niblack algorithm. (e) Output of the Otsu algorithm.

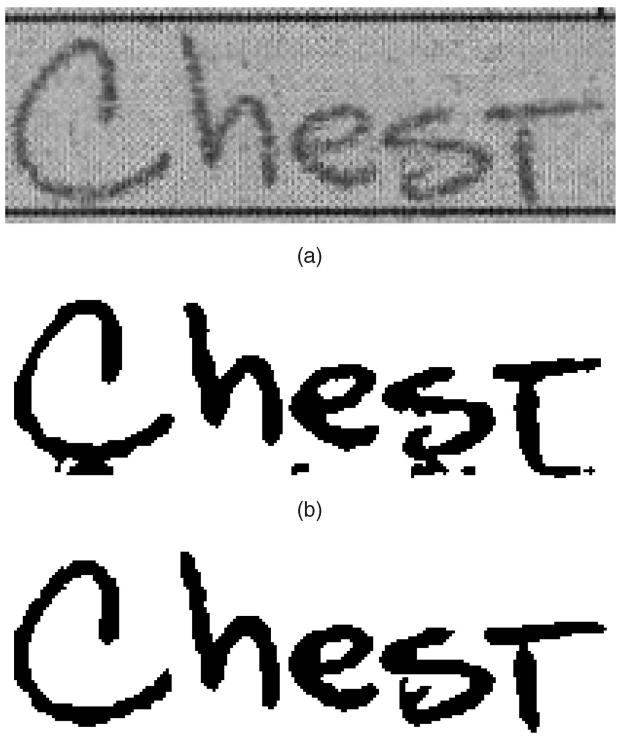


Fig. 12. Comparison of line removal results of the Milewski/Govindaraju algorithm and the MRF-based algorithm. (a) Input image. (b) Output of the Milewski/Govindaraju algorithm. (c) Output of the MRF-based algorithm.

We have also run the MRF binarization algorithm on some images from IAM DB3.0 [13]. We generated zero-mean Gaussian noise with deviations $\sigma = 50$, 70, and 100 in the IAM images to test the performance of binarization algorithms at different noise levels. For the group of images of $\sigma = 100$, a 3×3 mean filter was applied to all the images before binarization. The top-one word recognition rates of the original images and the images with Gaussian noise binarized by the MRF-based method, the Niblack algorithm, and the Otsu algorithm are shown in Table 4. Each group has 135 word images. We use a lexicon of 59 English words. The word recognition rates of the original images

TABLE 3
Comparison of Word Recognition Rates of the Milewski/Govindaraju Algorithm, the MRF-Based Approach, the Niblack Algorithm, and the Otsu Algorithm

Method		Milewski	**MRF**	Niblack	Otsu
Set #1	Top 1 rate	17.5%	**25.9%**	19.4%	11.6%
	Top 2 rate	24.4%	**36.6%**	26.9%	16.0%
	Top 5 rate	33.4%	**44.9%**	35.9%	23.3%
	Top 10 rate	39.6%	**51.7%**	42.3%	28.8%
Set #2	Top 1 rate	19.5%	**30.3%**	NA	NA
	Top 2 rate	28.1%	**40.7%**	NA	NA
	Top 5 rate	37.6%	**52.7%**	NA	NA
	Top 10 rate	45.0%	**60.0%**	NA	NA
Overall	Top 1 rate	18.7%	**28.6%**	NA	NA
	Top 2 rate	26.7%	**39.1%**	NA	NA
	Top 5 rate	36.0%	**49.7%**	NA	NA
	Top 10 rate	42.9%	**56.8%**	NA	NA

Set #1: Sample Word Images Not Affected by Forms Lines; Set #2: Sample Word Images Affected by Forms Lines; Overall: Set #1+Set #2

TABLE 4
Comparison of Word Recognition Rates (Top-One Accuracies in Percentage) of the MRF-Based Method, the Niblack Algorithm, and the Otsu Algorithm on Images with Different Noise Levels

	Original images	Gaussian Noise ($\sigma = 50$)	Gaussian Noise ($\sigma = 70$)	Gaussian Noise ($\sigma = 100$) and 3×3 Mean Filter
MRF	**83.0%**	**70.3%**	**43.7%**	**48.1%**
Niblack	83.0%	60.7%	31.1%	38.5%
Otsu	82.2%	65.2%	37.0%	37.8%

among all three methods are very close. The MRF-based method shows higher recognition rates on the images with Gaussian noise.

6 Conclusions

In this paper, we have presented a novel method for binarizing degraded document images containing handwriting and removing preprinted form lines. Our method models a binarized objective image as an MRF. In our MRF model, we reduce the large search space of the prior model to a class of 114 representatives by VQ and learn the observation model directly from the input image. We also presented an effective method of pruning the search space of the MRF. Our work is the first attempt at applying a stochastic method to the preprocessing of degraded high-resolution handwritten documents. Our model is targeted toward document images and therefore may not handle large variations in illumination, complex backgrounds, and blurring that are common in video and scene text processing. We will investigate approaches to generalize our model to these applications in our future work.

Acknowledgments

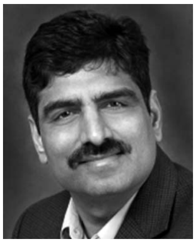
This work was supported by US National Science Foundation Grant 0429358.

References

[1] Z.-L. Bai and Q. Huo, "Underline Detection and Removal in a Document Image Using Multiple Strategies," *Proc. 17th Int'l Conf. Pattern Recognition,* vol. 2, pp. 578-581, 2004.

[2] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester, "Image Inpainting," *Computer Graphics, Proc. ACM SIGGRAPH '00,* pp. 417-424, 2000.
[3] H. Cao and V. Govindaraju, "Handwritten Carbon Form Preprocessing Based on Markov Random Field," *Proc. IEEE CS Conf. Computer Vision and Pattern Recognition,* 2007.
[4] W.T. Freeman and E.C. Pasztor, "Learning Low-Level Vision," *Proc. Seventh IEEE Int'l Conf. Computer Vision,* pp. 1182-1189, 1999.
[5] W.T. Freeman, E.C. Pasztor, and O.T. Carmichael, "Learning Low-Level Vision," *Int'l J. Computer Vision,* vol. 40, no. 1, pp. 25-47, 2000.
[6] S. Geman and D. Geman, "Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 6, no. 6, pp. 721-741, 1984.
[7] M.D. Gupta, S. Rajaram, N. Petrovic, and T.S. Huang, "Restoration and Recognition in a Loop," *Proc. IEEE CS Conf. Computer Vision and Pattern Recognition,* 2005.
[8] M.D. Gupta, S. Rajaram, N. Petrovic, and T.S. Huang, "Models for Patch Based Image Restoration," *Proc. Computer Vision and Pattern Recognition Conf.,* 2006.
[9] N.R. Howe, T.M. Rath, and R. Manmatha, "Boosted Decision Trees for Word Recognition in Handwritten Document Retrievals," *Proc. ACM SIGIR '05,* pp. 377-383, 2005.
[10] N. Jojic, B.J. Frey, and A. Kannan, "Epitomic Analysis of Appearance and Shape," *Proc. Ninth IEEE Int'l Conf. Computer Vision,* 2003.
[11] M. Kamel and A. Zhao, "Extraction of Binary Characters/Graphics Images from Grayscale Document Images," *CVGIP: Graphic Models Image Processing,* vol. 55, no. 3, 1993.
[12] G. Kim and V. Govindaraju, "A Lexicon Driven Approach to Handwritten Word Recognition for Real-Time Applications," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 19, no. 4, pp. 366-379, Apr. 1997.
[13] U. Marti and H. Bunke, "The IAM-Database: An English Sentence Database for Off-Line Handwriting Recognition," *Int'l J. Document Analysis and Recognition,* vol. 5, pp. 39-46, 2006.
[14] R. Milewski and V. Govindaraju, "Extraction of Handwritten Text from Carbon Copy Medical Form Images," *Proc. Seventh Int'l Workshop Document Analysis Systems,* pp. 106-116, 2006.
[15] W. Niblack, *An Introduction to Digital Image Processing.* Prentice Hall, 1986.
[16] N.A. Otsu, "A Threshold Selection Method from Gray-Level Histogram," *IEEE Trans. Systems, Man, and Cybernetics,* vol. 9, no. 1, 1979.
[17] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann, 1988.
[18] J. Sauvola, T. Seppanen, S. Haapakoski, and M. Pietiktinen, "Adaptive Document Binarization," *Proc. Fourth Int'l Conf. Document Analysis and Recognition,* pp. 147-152, 1997.
[19] M. Seeger and C. Dance, "Binarising Camera Images for OCR," *Proc. Sixth Int'l Conf. Document Analysis and Recognition,* pp. 54-58, 2001.
[20] C. Wolf and D. Doermann, "Binarization of Low Quality Text Using a Markov Random Field Model," *Proc. 16th Int'l Conf. Pattern Recognition,* 2002.
[21] Y. Yang and H. Yan, "An Adaptive Logical Method for Binarization of Degraded Document Images," *Pattern Recognition,* pp. 787-807, 2000.
[22] M. Yasuda, J. Ohkubo, and K. Tanaka, "Digital Image Inpainting Based on Markov Random Field," *Proc. Int'l Conf. Computational Intelligence for Modelling, Control and Automation and Int'l Conf. Intelligent Agents, Web Technologies and Internet Commerce,* vol. 2, pp. 747-752, 2005.
[23] J.-Y. Yoo, M.-K. Kim, S.Y. Han, and Y.-B. Kwon, "Line Removal and Restoration of Handwritten Characters on the Form Documents," *Proc. Fourth Int'l Conf. Document Analysis and Recognition,* pp. 128-131, 1997.

Huaigu Cao received the bachelor's and master's degrees from the Electronic Engineering Department, Tsinghua University, Beijing, in 2001 and 2004, respectively. He is a PhD student in the Department of Computer Science and Engineering at the University at Buffalo, State University of New York (SUNY). His research interests include image processing and pattern recognition. He is a member of the IEEE.



Venu Govindaraju received the BTech (Hons) degree from the Indian Institute of Technology (IIT), Kharagpur, India, in 1986, and the PhD degree from the University at Buffalo, State University of New York (SUNY), in 1992. He is a professor of computer science and engineering at the University at Buffalo, State University of New York (SUNY). In a research career spanning more than 20 years, he has made significant contributions to the areas of pattern recognition such as document analysis and biometrics. He has authored more than 245 scientific papers, including 45 journal papers and 20 book chapters. He has been the PI/co-PI of projects funded by the government and the industry for about 50 million dollars in the last 15 years. He is a fellow of the IEEE and the International Association of Pattern Recognition (IAPR).