Autonomous Document Cleaning—A Generative Approach to Reconstruct Strongly Corrupted Scanned Texts

Zhenwen Dai, *Member, IEEE* and Jörg Lücke, *Member, IEEE*

Abstract—We study the task of cleaning scanned text documents that are strongly corrupted by dirt such as manual line strokes, spilled ink, etc. We aim at autonomously removing such corruptions from a single letter-size page based only on the information the page contains. Our approach first learns character representations from document patches without supervision. For learning, we use a probabilistic generative model parameterizing pattern features, their planar arrangements and their variances. The model's latent variables describe pattern position and class, and feature occurrences. Model parameters are efficiently inferred using a truncated variational EM approach. Based on the learned representation, a clean document can be recovered by identifying, for each patch, pattern class and position while a quality measure allows for discrimination between character and non-character patterns. For a full Latin alphabet we found that a single page does not contain sufficiently many character examples. However, even if heavily corrupted by dirt, we show that a page containing a lower number of character types can efficiently and autonomously be cleaned solely based on the structural regularity of the characters it contains. In different example applications with different alphabets, we demonstrate and discuss the effectiveness, efficiency and generality of the approach.

Index Terms—Probabilistic generative models, document cleaning, scanned text, unsupervised learning, expectation maximization, variational approximation, expectation truncation

1 Introduction

A basic form of human communication, written text, consists of planar arrangements of reoccurring and regular patterns. While in modern forms of text these patterns are characters or symbols for words (e.g., Chinese texts), early forms consisted of symbols resembling objects. Written text became a successful form of communication because it exploits the readily available capability of the human visual system to learn and recognize regular patterns in visual data. In recent years, computer vision and machine learning became increasingly successful in analyzing visual data. Much progress has been made, for instance, by probabilistic modeling approaches that aim at capturing the statistical regularities of a given data set. Examples are image denoising by Markov Random Fields [1], [2], [3] or sparse coding models [4], [5], [6]. For many types of data, modeling approaches hereby have to address the problem that regular visual structures often appear at arbitrary positions. Sparse coding approaches indirectly address this problem by replicating a learned structure (e.g., a Gabor wavelet) at different positions of images. Other approaches go one step further and explicitly model pattern positions using additional hidden variables [7], [8], [9], [10], [11]. Such approaches allow for representations of patterns that are independent of their spatial positions. However, the combinatorics of pattern identity and position introduces major computational challenges because for each pattern class all positions ideally have to be considered.

In this paper we apply a probabilistic generative approach with explicit position encoding to clean strongly corrupted scanned text documents. The principle idea is very straight-forward: If characters are the salient regular patterns of text, an appropriately structured probabilistic model should be able to learn character representations as regular arrangements of features. In contrast, dirt is much more irregular. Coffee spots, spilled ink, or line-strokes scratching-out text share similar features with printed characters but such corruptions are, on average, much more random combinations of feature patterns. Based on this observation, the autonomous identification and recovery of characters from a corrupted text document should thus be possible. But how difficult is such a task? Or how robust can a solution of such a task be if the data is heavily corrupted by dirt? Would the information contained on a single page of a dirty document, for instance, be sufficient to identify the characters containing it? And if yes, can this be used to 'self-clean' the document? Such questions can, of course, not be answered by a clear 'yes' or 'no' because they will, e.g., depend on the type and degree of dirt or on the amount of available character information on a page. However, we will show that a self-cleaning of heavily corrupted documents is, indeed, possible, e.g., for relatively

- *Z. Dai is with the Department of Computer Science, University of Sheffield, Sheffield, South Yorkshire, United Kingdom. E-mail: z.dai@sheffield.ac.uk.*
- *J. Lücke is with the Cluster of Excellence Hearing4all and the School of Medicine and Health Sciences, University of Oldenburg, Oldenburg, Germany, and with the Department of Electrical Engineering and Computer Science, Technical University Berlin, Berlin, Germany. E-mail: joerg.luecke@uni-oldenburg.de.*

Manuscript received 29 Nov. 2012; revised 4 Feb. 2014; accepted 14 Feb. 2014. Date of publication 23 Mar. 2014; date of current version 10 Sept. 2014. Recommended for acceptance by M.S. Brown. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below. Digital Object Identifier no. 10.1109/TPAMI.2014.2313126

low numbers of different character types. The only prerequisite will hereby be the characters' regular feature arrangements. No information about the character shapes has to be available, which makes the approach applicable to entirely unknown character types. The problem addressed here is thus very different from the one aimed at by optical character recognition (OCR) methods that use supervised pretraining on known characters [12].

The idea of restoring scanned documents by removing corruptions and degradations using statistical information has previously been explored in literature. In the following we briefly review related earlier contributions: Markov source models (e.g., [13], [14]) have been proposed and applied for learning character representations, modeling relative locations among characters and restoring documents. An approach by Bern and Goldberg (2000) [15] clusters instances of the same symbol and computes super-resolved representatives for improving document images. A method by Zheng and Kanungo (2001) [16] estimates the parameters of a degradation model and constructs a lookup table for restoring the degraded image. Both of these two approaches [15], [16] focus on sub-regions of characters and are, e.g., applicable to line stroke corruptions. Recent work by Likforman et al. (2011) [43] combines total variation regularization and non-local means filtering for enhancing historical printed document images. Another recent approach by Moghaddam and Cheriet (2011) [17] divides a document image into a collection of patches, which are then individually corrected based on similar patches, before the corrected patches are used to build up a restored image; and work by Benerjee et al. (2009) [18] defines a Markov Random Field over sub-regions of characters, and restores a corrupted document image via a maximum a-priori (MAP) estimate.

Statistical models of scanned text documents can exploit statistical properties at different scales. Approaches focusing on small scale regularities are relatively general. For instance, sparse coding approaches can learn dictionaries of image patches in order to remove noise such as speckle noise, etc. [6]. However, for more structured noise, which shares features with the characters itself, more of the statistical structure of written text has to be captured. Models that focus on character sub-regions (e.g., [15], [16], [18]) can remove more structured noise such as line strokes that are sufficiently dissimilar to character parts [15], [16]. The approach presented in this paper goes one step further by statistically learning to represent whole characters as planar relations of pattern features. Such a higher-level character representation can allow for a removal of corruptions even if the sub-regions of corrupting patterns (line strokes) are similar to character sub-regions. At the same time, corruptions such as speckle noise or cuts and breaks in historical documents can be removed. However, the larger the scale (the patch size) and complexity of the statistical model, the more challenging the inference and training problem becomes. Larger scale regularities other than representations of whole characters can also be captured. The approaches by Kopec and Chou [13], [14] learn whole characters and exploit the statistical property of text, but their learning algorithm requires the transcriptions of target documents (supervision information), which is different from the unsupervised approach followed here.

The generative model we apply for modeling character patterns is similar to models for visual objects suggested by Williams and Titsias [11], [19] and Jojic and Frey [20] (known as *sprite models*). Sprite models are generative models of visual scenes allowing for arbitrary planar positions of objects but they assume a fixed order in depth (an object always has the same distance from the camera). With such models, explicit representations of objects can be learned from data. Many extensions have been made to further enhance the model, e.g., with a deformable model of objects [21], [22], allowing affine transformations [23], and speeding up with various techniques [24]. Sprite models have also been suggested for video layer decomposition and optical flow, where the accuracy of segmentation and motion estimation are of importance, e.g., [25], [26], [27], [28]. Besides layered models, epitomic image analysis [29] is another related generative approach, which, instead of building explicit object representations, summarizes an image into a miniature, condensed version containing the essence of the textural and shape properties of the image. The extracted epitome can be used for different inference tasks such as image segmentation, motion estimation as well as location recognition [30] (after post-processing). As the data points we will have to process are image patches of corrupted text documents, these previous models are not applicable because they require a static background, do not provide a mechanism to discriminate characters from irregular patterns, and are based on pixel image representations which can make learning less robust. In contrast, we (1) will have to allow for varying fore- and background patterns (to take corruptions into account), (2) will introduce a mechanism for character versus non-character discrimination, and (3) will consider general feature vector representations of the data. Together with a novel non-greedy training scheme in the form of truncated variational EM [31], the derived method will provide the required robustness and efficiency for the task.

2 A Probabilistic Generative Model for Characters

The probabilistic model we consider generates small image patches of size $\vec{D} = (D_1, D_2)$. A pixel at position $\vec{d}$ of the patch is represented by a feature vector $\vec{y}_{\vec{d}}$ with F entries. For now $\vec{y}_{\vec{d}}$ can be thought of as a color vector at pixel position $\vec{d}$ in RGB space ($F = 3$). Later on, we will use higher-dimensional feature vectors that more robustly encode local image information. A patch $Y = (\vec{y}_{(1,1)}, \ldots, \vec{y}_{(D_1,D_2)})$ is modeled to contain one pattern of class c at an arbitrary position $\vec{x}$ of the patch. The generative model is introduced step-by-step below, Fig. 1 shows its corresponding graphical model, and Fig. 2 illustrates an example of patch generation.

For the generation of a patch Y, we first choose a pattern class c using a standard mixture model with $\vec{\pi} = (\pi_1, \ldots, \pi_C)$ denoting the mixing proportions and with C denoting the total number of classes:

$$p(c \,|\, \vec{\pi}) = \pi_c \quad \text{with} \quad \sum_{c=1}^{C} \pi_c = 1\,. \tag{1}$$

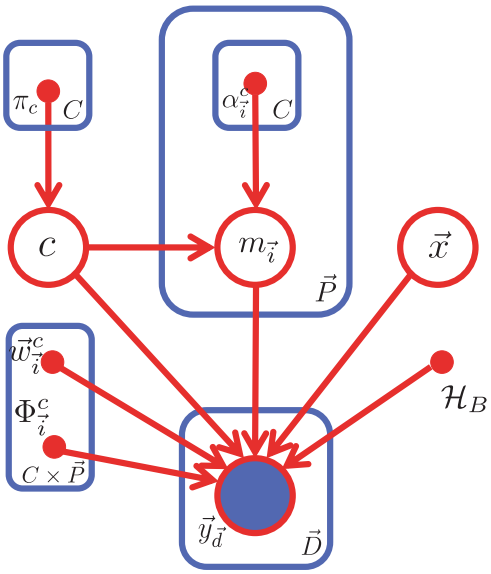


Fig. 1. Graphical representation of the generative model.

The pattern position within the patch, $\vec{x} \in \mathcal{D}, \mathcal{D} = \{1, \ldots, D_1\} \times \{1, \ldots, D_2\}$, is a 2D vector which is chosen independently of the class from a uniform distribution over the entire patch:

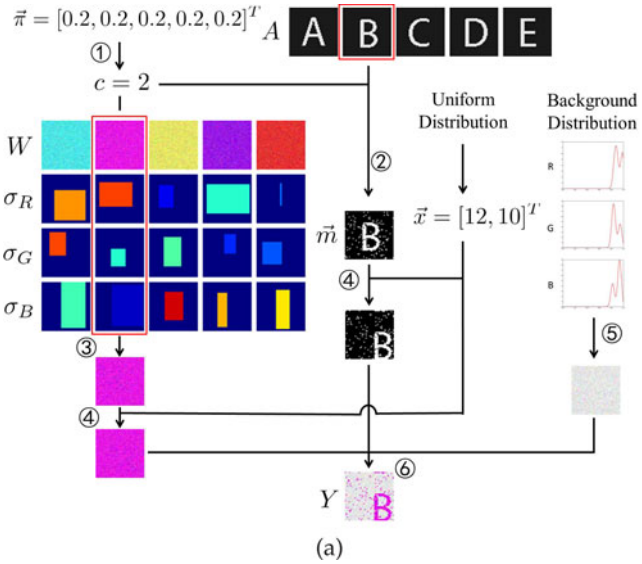
$$p(\vec{x}) = p(x_1)p(x_2) = \text{Uniform}(1, D_1) \times \text{Uniform}(1, D_2) = \frac{1}{D_1 D_2}. \quad (2)$$

The shapes of different patterns are modeled by a set of binary latent variables, namely the pattern *mask*: $\vec{m} = (m_{(1,1)}, \ldots, m_{(P_1,P_2)})$, where $m_{\vec{i}} \in \{0, 1\}$. For a value $m_{\vec{i}} = 1$, a feature of the corresponding pattern is chosen as the generated feature at position $\vec{i}$, while for $m_{\vec{i}} = 0$, the feature is chosen from a background distribution. The pattern size $\vec{P} = (P_1, P_2)$ can hereby be different from the image patch size $P_1 \leq D_1, P_2 \leq D_2$ to account for the fact that the patches will be chosen significantly larger than the characters they contain. Given a pattern class c, the mask variables are drawn from Bernoulli distributions:

$$p(\vec{m} \mid c, A) = \prod_{\vec{i}=(1,1)}^{(P_1,P_2)} p(m_{\vec{i}} \mid c, A) = \prod_{\vec{i}=(1,1)}^{(P_1,P_2)} \left(\alpha^c_{\vec{i}}\right)^{m_{\vec{i}}} \left(1 - \alpha^c_{\vec{i}}\right)^{1-m_{\vec{i}}}, \quad (3)$$

where $A = (A^1, \ldots, A^C)$ with $A^c = (\alpha^c_{(1,1)}, \ldots, \alpha^c_{(P_1,P_2)})$ are the parameters of the mask distribution. For the area where the image patch is outside the pattern, the mask variables are always assigned to zero: $p(m_{\vec{i}} = 1 \mid c, A) = p(m_{\vec{i}} = 1) = 0$ $\forall \vec{i} \notin \mathcal{P}$ with $\mathcal{P} = \{1, \ldots, P_1\} \times \{1, \ldots, P_2\}$.

From the definition of masks, a background distribution is required for all those features not belonging to a pattern ($m_{\vec{i}} = 0$). A possible choice is a flat Gaussian distribution (compare [11]). However, for data such as patches from corrupted text documents, the distribution values are often very different for the different feature vector entries, and for the dirty background are often observed to be non-Gaussian. To appropriately model the background features, we therefore construct a probability density function $\mathcal{H}_B$ by computing the histogram of different feature values across the image patches. The probability densities for the individual feature vector entries will be modeled individually (see Fig. 4a for histograms of R, G, and B channel). The histograms are computed across all the image patches including the features that are potentially later identified as being part of the learned patterns. Nevertheless, the computed histograms are usually very similar to the true background distributions (compare Fig. 4a). Once computed we therefore leave the histograms fixed throughout learning. Having defined the background distribution $\mathcal{H}_B$ and given pattern class c,



1) A pattern class is chosen, e.g., the class with pattern "B".
2) The mask variable $\vec{m}$ is drawn for the distribution of the chosen class (3).
3) The internal representation of the pattern (without background) is sampled from the corresponding Gaussian distribution.
4) The pattern parameters and mask variable are translated by a random position $\vec{x}$ drawn from a uniform distribution.
5) The background is sampled from the histogram distribution of background.
6) The final image patch is generated by combining the internal representation of the pattern with the background according to the mask variable.

(b)

Fig. 2. (a) An illustration of the generation process. (b) Summary of the generation process.

mask $\vec{m}$, and pattern position $\vec{x}$, the distribution of patch features is given by

$$p(Y \,|\, c, \vec{m}, \vec{x}, \Theta) = \prod_{\vec{d}=(1,1)}^{(D_1,D_2)} \Big[m_{(\vec{d}-\vec{x})} \mathcal{N}\Big(\vec{y}_{\vec{d}}\,; \vec{w}^{\,c}_{(\vec{d}-\vec{x})}, \Phi^{c}_{(\vec{d}-\vec{x})}\Big) + (1 - m_{(\vec{d}-\vec{x})}) \mathcal{H}_B(\vec{y}_{\vec{d}}) \Big], \tag{4}$$

where $\vec{w}^{\,c}_{\vec{i}}$ is the mean of the Gaussian distribution and $\Phi^{c}_{\vec{i}}$ is the diagonal convariance matrix: $\Phi^{c}_{\vec{i}} = \mathrm{diag}((\sigma^{c}_{\vec{i},f=1})^2, \ldots, (\sigma^{c}_{\vec{i},f=F})^2)$. The mean $\vec{w}^{\,c}_{\vec{i}}$ parameterizes the mean feature vector of pattern c at position $\vec{i}$ relative to the pattern position $\vec{x}$. The variance vector $\Phi^{c}_{\vec{i}}$ parameterizes the feature vector variances (different variance per vector entry). The shift of a pattern c is implemented by a change of the position indices $\vec{i}$ by $\vec{x}$ using cyclic boundary positions:

$$\begin{aligned} \vec{d} &= (\vec{i} + \vec{x}) \\ &:= ((i_1 + x_1) \bmod D_1, (i_2 + x_2) \bmod D_2)^T. \end{aligned} \tag{5}$$

The cyclic boundary condition is used mainly for computational convenience. Otherwise, the search space for translations will increase dramatically (about eight times bigger), which will significantly increase the computational cost.

Equations (1) to (5) define the generative model for image patches. The parameters of the model are given by $\Theta = (W, \Phi, A, \vec{\pi})$ with $W = (W^0, \ldots, W^C)$ and $W^c = (\vec{w}^{\,c}_{(1,1)}, \ldots, \vec{w}^{\,c}_{(P_1,P_2)})$, and together with the histograms for the background distribution. Fig. 2 shows schematically how a patch is generated for a given set of parameters.

3 Efficient Likelihood Maximization

For a given set of image patches $\mathcal{Y} = (Y^{(1)}, \ldots, Y^{(N)})$ we seek the parameters that best model the data set. One approach of learning the parameters is to maximize the data likelihood:

$$\begin{aligned} \Theta^* &= \arg\max_{\Theta}\{\mathcal{L}(\Theta)\}, \\ \mathcal{L}(\Theta) &= \log\big(p(Y^{(1)}, \ldots, Y^{(N)} \,|\, \Theta)\big). \end{aligned} \tag{6}$$

A frequently used method to find the parameters Θ^* is Expectation Maximization (EM), which iteratively optimizes a lower bound of the likelihood $\mathcal{F}(\Theta, q)$ w.r.t. the parameters Θ and a distribution q. Given the data and the current model parameters Θ, q is an approximation to the posterior distribution over the hidden variables [32]. With $\sum_V$ denoting a summation across the joint space of all hidden variables $V = (c, \vec{m}, \vec{x})$ the lower-bound is given by

$$\begin{aligned} \mathcal{F}(\Theta, q) = &\sum_{n=1}^{N} \sum_{V} q_n(V, \Theta') \log\left(p(Y^{(n)}, V \,|\, \Theta)\right) \\ &- \sum_{n=1}^{N} \sum_{V} q_n(V, \Theta') \log\left(q_n(V, \Theta')\right), \end{aligned} \tag{7}$$

where Θ' denotes the parameters from the previous iteration.[1]

M-step. Parameter update rules are canonically derived by setting the derivatives of $\mathcal{F}$ w.r.t. the parameters to zero. For the model (1)-(5), we obtain

$$\pi_c = \frac{1}{N} \sum_n \sum_{\vec{x}} p^{(n)}_{\Theta}(c, \vec{x}), \tag{8}$$

$$\alpha^{c}_{\vec{i}} = \frac{\sum_n \sum_{\vec{x}} p^{(n)}_{\Theta}(c, \vec{x}) p^{(n)}_{\Theta}(m_{\vec{i}} = 1 \,|\, c, \vec{x})}{\sum_n \sum_{\vec{x}} p^{(n)}_{\Theta}(c, \vec{x})}, \tag{9}$$

$$\vec{w}^{\,c}_{\vec{i}} = \frac{\sum_n \sum_{\vec{x}} p^{(n)}_{\Theta}(c, \vec{x}) p^{(n)}_{\Theta}(m_{\vec{i}} = 1 \,|\, c, \vec{x}) \vec{y}^{\,(n)}_{(\vec{i}+\vec{x})}}{\sum_n \sum_{\vec{x}} p^{(n)}_{\Theta}(c, \vec{x}) p^{(n)}_{\Theta}(m_{\vec{i}} = 1 \,|\, c, \vec{x})}, \tag{10}$$

$$\begin{aligned} \Phi^{c}_{\vec{i}} = &\frac{1}{\sum_n \sum_{\vec{x}} p^{(n)}_{\Theta}(c, \vec{x}) p^{(n)}_{\Theta}(m_{\vec{i}} = 1 \,|\, c, \vec{x})} \sum_n \sum_{\vec{x}} p^{(n)}_{\Theta}(c, \vec{x}) \\ &\times p^{(n)}_{\Theta}(m_{\vec{i}} = 1 \,|\, c, \vec{x}) \big(\vec{w}^{\,c}_{\vec{i}} - \vec{y}^{\,(n)}_{\vec{d}}\big) \big(\vec{w}^{\,c}_{\vec{i}} - \vec{y}^{\,(n)}_{\vec{d}}\big)^T \odot \mathbb{1}, \end{aligned} \tag{11}$$

where we use the abbreviations: $p^{(n)}_{\Theta}(m_{\vec{i}} | \, c, \vec{x}) := p(m_{\vec{i}} | \, c, \vec{x}, Y^{(n)}, \Theta)$, $p^{(n)}_{\Theta}(c, \vec{x}) := p(c, \vec{x} \,|\, Y^{(n)}, \Theta)$, and where $\odot$ denotes pointwise matrix multiplication (in this case with the unit matrix $\mathbb{1}$). For the derivations of the M-step equations we refer to Appendix A, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPAMI.2014.2313126.

E-step. The crucial and computationally expensive part of EM is the computation of the expectation values w.r.t. the posterior. For each data point, this involves summations of probabilities for all combinatorics of the hidden variables c, $\vec{m}$ and $\vec{x}$. However, the combinatorics can be simplified. By exploiting the standard assumption of independent observed variables (compare, e.g., [4], [5]) given the latents in (4), the posterior distribution over $\vec{m}$ can be decomposed into a product of the posteriors over individual binary masks as follows:

$$p(c, \vec{m}, \vec{x} \,|\, Y, \Theta) = \left(\prod_{\vec{i}=(1,1)}^{(P_1,P_2)} p(m_{\vec{i}} | \, c, \vec{x}, Y, \Theta) \right) p(c, \vec{x} \,|\, Y, \Theta). \tag{12}$$

The posterior distribution over individual binary masks can then be computed according to

$$p(m_{\vec{i}} \mid c, \vec{x}, Y, \Theta) = \frac{p(\vec{y}_{(\vec{i}+\vec{x})}, m_{\vec{i}} \mid c, \vec{x}, \Theta)}{\sum_{m'_{\vec{i}}} p(\vec{y}_{(\vec{i}+\vec{x})}, m'_{\vec{i}} \mid c, \vec{x}, \Theta)}. \tag{13}$$

The summation in the denominator can be computed efficiently as it only contains two cases: $m_{\vec{i}} = 0$ and $m_{\vec{i}} = 1$. The posterior distribution over c and $\vec{x}$ can be computed as follows:

1. As, in the following text, only the notation of the parameters from previous iteration has been used, we omit the notation Θ' and use Θ to indicate the parameters from previous iteration.

$$p(c,\vec{x}\,|\,Y,\Theta) \propto \left[\prod_{\vec{i}=(1,1)}^{(P_1,P_2)} \left(\sum_{m_{\vec{i}}} p(\vec{y}_{(\vec{i}+\vec{x})}, m_{\vec{i}}\,|\,c,\vec{x},\Theta)\right)\right] \times p(\vec{x}\,|\,\Theta)p(c\,|\,\Theta). \tag{14}$$

With such a decomposition (compare [11]), the computational complexity decreases from exponential to polynomial, which makes the computation tractable in principle. However, the computational complexity still grows very fast with the size of patterns and image patches, $\mathcal{O}(CD_1D_2P_1P_2)$. For realistic image sizes (e.g., usually hundreds of thousands of pixels), it still exceeds currently available computational resources.

To further improve efficiency, we therefore approximate the computation of expectation values using variational EM (e.g., [33]). Source of the large computation is the demand of evaluating all possible pattern positions for all classes. To reduce the number of hidden states that have to be evaluated, we apply a recent variational EM approach (Expectation Truncation, [31]) which is directly applicable to discrete hidden variables. The used approach is not based on the usual factored form of q but on a truncated variational approximation to the posterior. Applied to the posterior in (14), it is given by

$$p\big(c,\vec{x}\,|\,Y^{(n)},\Theta\big) \approx q_n(c,\vec{x};\Theta) = \frac{p(c,\vec{x},Y^{(n)}\,|\,\Theta)}{\sum_{(c,\vec{x})\in\mathcal{K}_n} p(c,\vec{x},Y^{(n)}\,|\,\Theta)} \;\forall (c,\vec{x}) \in \mathcal{K}_n, \tag{15}$$

and zero otherwise. The variational distribution q_n approximates the true posterior with high precision if the set $\mathcal{K}_n$ contains those classes and positions that carry most posterior mass for a given data point $Y^{(n)}$. This means that we have to find, for each patch, the most likely pattern classes together with their most likely positions in order to obtain a high quality approximation. Therefore, we define a function $\mathcal{S}^{(n)}_{\Theta}(c,\vec{x})$ that assigns a score to each class and position pair $(c,\vec{x})$:

$$\mathcal{S}^{(n)}_{\Theta}(c,\vec{x}) = \prod_{\vec{i}\in\mathcal{P}'_c} \big[\mathcal{N}\big(\vec{y}^{(n)}_{(\vec{i}+\vec{x})}\,;\vec{w}^c_{\vec{i}},\Phi^c_{\vec{i}}\big)p(m_{\vec{i}}=1\,|\,\Theta) + \mathcal{H}_B\big(\vec{y}^{(n)}_{(\vec{i}+\vec{x})}\big)p(m_{\vec{i}}=0\,|\,\Theta)\big]p(\vec{x}\;|\;\Theta)p(c\,|\,\Theta), \tag{16}$$

with $\mathcal{P}'_c \subseteq \mathcal{P}$. This scoring (or selection) function (compare [31]) gives high values to all those positions that are consistent with features in the set $\mathcal{P}'_c$. The set $\mathcal{P}'_c$ is in turn defined to contain the λ most reliable features of pattern c. We define these features as those with the highest mask parameters $\alpha^c_{\vec{i}}$. A small λ results in a very efficiently computable function $\mathcal{S}^{(n)}_{\Theta}(c,\vec{x})$. Based on the selection function, we now define the set of the most likely class and position pairs to be

$$\mathcal{K}_n = \{(c,\vec{x})\,|\,(c,\vec{x}) \text{ has one of the } (K\,C\,D_1\,D_2) \text{ largest values of } \mathcal{S}^{(n)}_{c,\Theta}(\vec{x})\}, \tag{17}$$

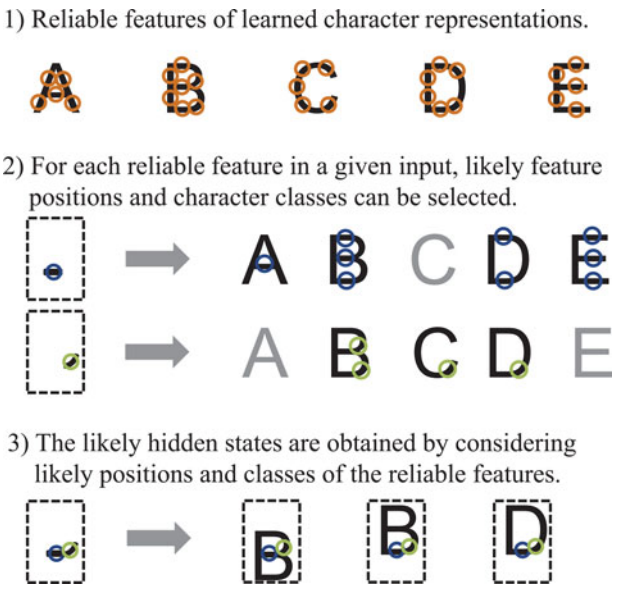


Fig. 3. An illustration of the applied selection process. 1) Reliable features can be defined based on the learned character representations (red circles). 2) If such a feature is present in a given input, the most likely classes and positions can be selected (different such classes and position pairs for each feature). 3) The most likely classes and positions for each feature can be combined to select a finally small number of possible class and position pairs. Note that the selection process has been simplified to communicate the basic idea. The actual selection behaves probabilistically by ranking the configurations of the posterior distribution according to its scores instead of making deterministic decisions as shown here.

where $K \in [0,1]$ is the fraction of the joint space of all classes and positions (with size CD_1D_2).

Note that, as shown in some examples of image patches in Fig. 5a, usually only a small number of characters are present in a patch compared to the total number of possible characters. Therefore, given an image patch, the probabilistic mass in its posterior distribution will be concentrated in small volumes of the joint hidden space for c and $\vec{x}$. A prerequisite for the applicability of the truncated variational approach [31] is thus fulfilled. To efficiently find the places with high concentration of posterior mass, which is the goal of the selection function $\mathcal{S}^{(n)}_{\Theta}(c,\vec{x})$, we exploit that often very good guesses about an objects identity can be made based on partially observed information. The selection function in (16) is defined to preselect pattern classes and positions based on few but reliable pattern features. The features are themselves depending on the model parameters and evolve during learning. The selection of regions $\mathcal{K}_n$ can hereby be generous as selected hidden states without high probability mass do not negatively effect the approximation (except of increased computational demand). An illustrative example of the selection process is given in Fig. 3.

Note that the principle idea of efficient inference through preselection has been proposed and discussed in different contexts before [34], [35], [36], [37] including character perception [38]. In the context of probabilistic visual inference Yuille and Kersten [37] have abstractly discussed the idea using the example of character patterns. Approximate inference by preselection was then shown to correspond to a variational Bayesian EM approach (Expectation Truncation,

TABLE 1
Summary of the Learning Algorithm

choose approximation parameters $K = 0.02$ and $\lambda = 200$
initialize features W with values of a pattern size region at a random position of a randomly selected patch
initialize feature variances Φ with the standard deviation of the data set (each feature dimension individually)
initilize mask parameters A with values uniformly drawn from the interval $[0, 1]$
while the parameters have not converged **do**
  select the representative feature index set $\mathcal{P}'_c$ for each pattern c
  for each data point $n = 1, \ldots, N$ **do**
    compute $\mathcal{K}_n$ by evaluating $\mathcal{S}^{(n)}_\Theta(c, \vec{x})$ on the joint space of c and $\vec{x}$ using (16) and (17)
    compute the approximate posterior distribution (15) over the truncated set $\mathcal{K}_n$
  end for
  update the parameters in the M-step (11) using the approximate posterior distributions
end while

[31]) which allowed for concrete derivations of efficient inference and learning algorithms for generative models. While the approach has successfully been applied for sparse coding models (e.g., [31], [39]), the selection as used for our model closely corresponds to the abstract example of inference for characters by Yuille and Kersten [37]. Using the function (16), efficient selection is achieved by only checking for the most reliable features of each pattern. Reliability is hereby measured based on the mask parameters: only the λ features associated with the highest values of the mask parameters are considered (see $\mathcal{P}'_c$ in (16)).

In principle, the approximation [31] can also be used to constrain the number of states of mask variables. However, the computational gain is negligible as the posterior w.r.t. the mask can be computed efficiently (13). For the approximation, note that λ and K parameterize the accuracy. The higher the value of λ is, the more reliable is the selection of considered classes and positions. The higher the value of K is, the larger is the considered area of the joint class and position space. However, the larger both the λ and K are, the higher is the computational cost. For the highest possible value of λ the selection becomes optimal in the sense that $\mathcal{S}^{(n)}_\Theta(c, \vec{x})$ becomes proportional to $p(c, \vec{x} \,|\, Y^{(n)}, \Theta)$.[2] For the highest possible value of K, $K = 1$, all positions are considered and the variational distribution (15) becomes equal to the exact posterior. In numerical experiments we observed that approximations with high accuracy and simultaneously low computational costs are obtained already for relatively low numbers of λ (e.g., $\lambda = 200$ out of $P_1 P_2$ features) and relatively low fractions of considered joint space (e.g., $K = 0.02$).

The overall procedure of the proposed learning algorithm is summarized in Table 1. For each EM iteration, the computation for each data point can be carried out independently until the intermediate results are summed together for the new parameters. Therefore, we parallelized the computation by partitioning data points, evenly distributing them across computer nodes/cores, and gathering intermediate results at the end of each EM iteration (see Appendix B, available in the online supplemental material).

2. $\mathcal{S}^{(n)}_\Theta(c, \vec{x})$ becomes equal to $p(c, \vec{x} \,|\, Y^{(n)}, \Theta) p(Y^{(n)} \,|\, \Theta)$ with $p(Y^{(n)} \,|\, \Theta)$ being a constant for the selection.

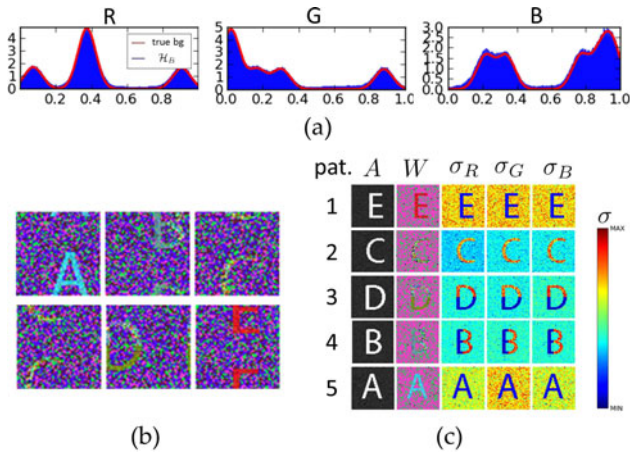


Fig. 4. Experiment on artificial data. (a) The background for data generation (red curves) and constructed background histogram $\mathcal{H}_B$ (blue regions). (b) Five samples of $N = 1{,}000$ generated image patches. (c) The learned model parameters. The algorithm terminated after 71 iterations. The mask parameter A is visualized in gray-scale images, the pattern feature W is visualized in RGB color space and the noise parameters σ_R, σ_G and σ_B are visualized by heat maps.

4 Learning Representations of Characters

Equations (6) to (17) define an approximate EM algorithm for learning character representations. It will be used to clean corrupted documents as described later. Before, we numerically evaluate the learning algorithm itself.

4.1 Artificial Data

Let us first consider artificial images for which ground truth information is available. For the training data, we generated $N = 1{,}000$ RGB image patches ($F = 3$) of size $\vec{D} = (50, 50)$ according to the model (1) to (5). Each patch contained one of five different character types with equal probability ($\pi_c = 0.2$). The chosen colored characters were generated from corresponding mask, mean and variance parameters (see Fig. 2). The background color was drawn from a Mixture of Gaussians as an example of multi-modal distributions (compare Fig. 2a). Fig. 4b shows a random selection of five generated data points. The derived EM learning algorithm was applied to the data assuming $C = 5$ classes and $\vec{P} = \vec{D} = (50, 50)$. First, the background histograms $\mathcal{H}_B$ was computed from the whole data set, and was observed to model the true generating RGB-distributions with high accuracy (the blue regions in Fig. 4a shows the learned histograms compared to the true distributions in red). The remaining model parameters were randomly initialized: the pattern mean W was independently and uniformly drawn from the RGB-color-cube $[0, 1]^3$; the pattern variance Φ was set to the standard deviation of the data set; and the initial mask parameters A were uniformly drawn from the interval $[0, 1]$.

The learning course of the parameters is illustrated in Fig. 4c with iteration 0 showing the initial values. After iteration 70, parameters had converged sufficiently. To visualize pattern variances in Fig. 4c, they were organized as a

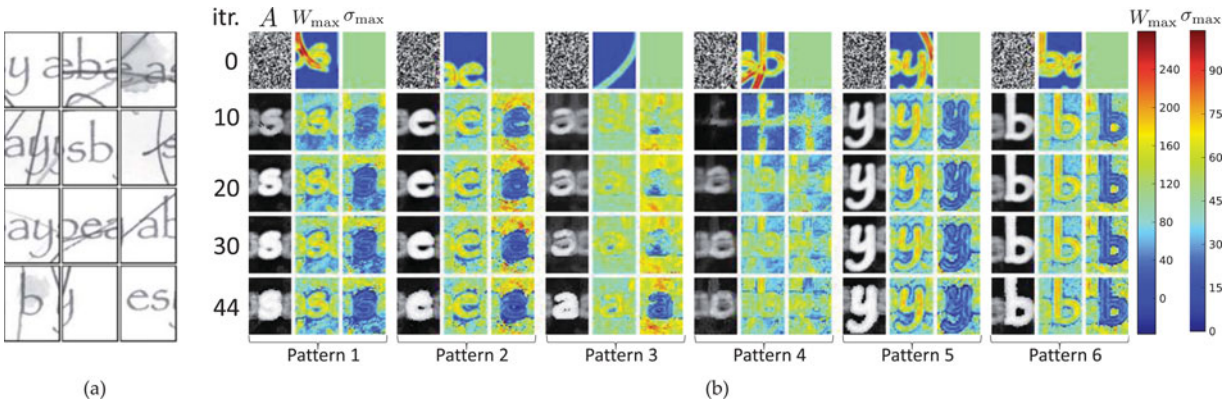


Fig. 5. Experiment on a scanned text document. (a) 12 samples of $N = 1{,}379$ image patches. (b) The learning course of the parameters. The mask parameter A is visualized in gray-scale images and the pattern feature W and the pattern noise parameter σ are visualized by heat maps. For W and σ we only show the maximal values, e.g., $W^c_{\vec{i},\max} = \max_f(W^c_{\vec{i},f})$, to enable a compact visualization.

matrix for each pattern and each feature dimension, e.g., $\sigma^c_R = (\sigma^c_{(1,1),f=R}, \ldots, \sigma^c_{(P_1,P_2),f=R})$. These variance matrices are visualized as color images which are normalized individually. As can be observed, the algorithm successfully learned the model parameters. For the experiment of Fig. 4 and other repeated experiments, the learned parameters diverged from the generating parameters by on average less than 3.0 percent (excluding the cases converging to local optima). Convergence to local optima has only been observed in very few cases (one out of 10 runs).

4.2 Scanned Text Documents

Let us now apply the learning algorithm to data from a single page of a scanned text document. Consider the corrupted document displayed in Fig. 8 (left-hand-side) which contains five character types, "a", "b", "e", "s" and "y". The printed document was manually corrupted with dirt in the form of line-strokes and with grayish spots. The data set for training was created by a high-resolution scan of the document ($3{,}307 \times 4{,}677$ pixels) and by automatically cutting the scan into small overlapping patches (120×165 pixels) with fixed patch distances. Fig. 5a shows some examples of such patches. The patch size was chosen to easily contain whole characters and patches were cut along the writing direction of the horizontally aligned text. White patches were automatically discarded via thresholding. While an appropriate working of the algorithm requires patch sizes large enough for character patterns, a cutting along writing directions is not required (see later discussions of experiments in Figs. 8 and 9). The cut-out patches are used to generate the actual data points $Y^{(n)}$ with vectorial features. Instead of RGB feature vectors as for the introductory example, we used feature vectors generated through Gabor filter responses. Gabor features are robust and widespread in image processing (see, e.g., [8], [40]) with high sensitivity to edge-like structures and textures. Furthermore, they are tolerant w.r.t. small local deformations and brightness changes. For the small patches we computed a Gabor feature with 40 entries (five scales and eight directions) at every third pixel [8], which resulted in 2D arrays of $D_1 \times D_2 = 40 \times 55$ Gabor feature vectors.

The learning algorithm was applied to this data set assuming $C = 6$ classes. The pattern mean W was initialized by randomly selecting C patches from the data set and cutting out a segment of the pattern size at random positions. The remaining parameters were initialized in the same way as for the artificial data. To increase computational efficiency we, furthermore, assumed with $\vec{P} = (30, 40)$ a pattern size smaller than the patch size but still larger than the size of any characters. Parameter optimization (44 EM iterations) took about 25 minutes on a cluster with 15 GPUs (Nvidia GTX 480). More implementation details about the algorithm's parallelization can be found in Appendix B, available in the online supplemental material.

Fig. 5b visualizes the time-course of the learning algorithm. As can be observed, the parameters converged to appropriately represent the five character types. They are represented by different mask parameters, mean features and feature variances of the different classes. As only five classes are required to represent all the characters, one class converged to an average of some patterns and dirt (see Pattern 4 in Fig. 5b). In numerical experiments on this and other documents, the classes not representing characters had either much lower values for learned mask parameters (compare Fig. 5b) or much lower values for learned mixing proportions π_c. We exploited this observation to automatically identify character classes (see Section 6.1 for details). In this way we further increase the robustness of the learning procedure by (1) repeating the learning algorithm multiple times with different randomly chosen initial conditions and (2) by selecting the parameters of a run with the highest number of character classes.

5 Character Detection and Identification

Based on a character representation learned as described above, characters in a given corrupted document can be detected and identified. We screen through the whole document from upper-left to lower-right patch by patch. Our first aim is to identify in each patch $Y^{(n)}$ the position and the class of the pattern most similar to that of a learned character. To identify the type and position of this best fitting pattern, we compute the maximum a-posteriori (MAP) estimate of the approximate posterior:

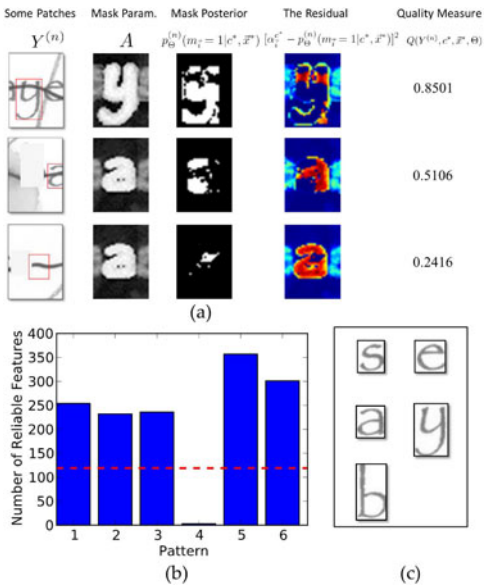


Fig. 6. (a) An illustration of the match quality. The first column shows the patches with the MAP estimate (red rectangle); the second column shows the mask parameters of the matched pattern class; the third column shows the corresponding posterior probability of the mask variables; the forth column shows the difference between the mask parameters and posterior; the fifth column states the resulting quality measure (see Section 5 for details). (b) Visualization of the number of reliable features for each pattern and the threshold used for selecting character representations (dashed red line). (c) The clean representations of each pattern with their bounding box for reconstruction.

$$(c^*, \vec{x}^*) = \arg \max_{(c,\vec{x}) \in \mathcal{K}_n} \{q_n(c, \vec{x}\,; \Theta)\} \approx \arg \max_{c,\vec{x}} \{p(c, \vec{x} \,|\, Y^{(n)}, \Theta)\}, \tag{18}$$

with $q_n(c, \vec{x}\,; \Theta)$ and $\mathcal{K}_n$ defined as in Section 3. In analogy to template matching ([8], [41] and many more) we refer to the result of the MAP estimate (18) as the *match* for the image patch, to $\vec{x}^*$ as the *matched position* and to c^* as the *matched class*.

As some of the patches may not contain any or any complete character pattern (see, e.g., Fig. 6a, left-hand-side), we introduce a quality measure to distinguish good matches (to characters) from bad matches (to non-characters). Given the patch $Y^{(n)}$ with match $(c^*, \vec{x}^*)$, we define the *quality* of the match as follows:

$$Q(c^*, \vec{x}^*, Y^{(n)}, \Theta) = 1 - \frac{\sum_{\vec{i}=(1,1)}^{(P_1,P_2)} \left(\alpha_{\vec{i}}^{c^*}\right)^{\gamma} \left[\alpha_{\vec{i}}^{c^*} - p\left(m_{\vec{i}} = 1 \,|\, c^*, \vec{x}^*, Y^{(n)}, \Theta\right)\right]^2}{\sum_{\vec{i}'=(1,1)}^{(P_1,P_2)} \left(\alpha_{\vec{i}'}^{c^*}\right)^{\gamma}}, \tag{19}$$

where $p(m_{\vec{i}} = 1 \,|\, c^*, \vec{x}^*, Y^{(n)}, \Theta)$ is the posterior distribution of the binary mask in (13). The negative term in (19) is a normalized distance measure between mask parameters and mask posterior probabilities. Low values of Q correspond to poor matches and $Q = 1$ corresponds to a perfect match.

The definition of the match *quality* follows an observation that, for good matches, a large part of the image patch is consistent with the corresponding character, while, for bad matches, only a small part is consistent. To convert such observation into a quantitative measure, we need to define this consistency. As can be seen in Fig. 6a, the consistency can be well formulated using the distance between the mask parameters and mask posterior probabilities. Intuitively speaking, the mask parameters show which features should be consistent in order to be considered as the pattern and the mask posterior probabilities show which features in the image patch are actually consistent with the pattern (see the second and third columns in Fig. 6a). To make the match quality $Q(c^*, \vec{x}^*, Y^{(n)}, \Theta)$ independent of the pattern size, we added normalization weights $(\alpha_{\vec{i}}^{c^*})^{\gamma}$. Besides this reason, we also noticed that, to determine a good match, it is crucial whether the reliable features ($\alpha_{\vec{i}}^{c^*}$ close to 1) are well matched ($p(m_{\vec{i}} = 1 \,|\, c, \vec{x}, Y^{(n)}, \Theta)$ close to 1), while other features are usually irrelevant. On the other hand, to be tolerant w.r.t. corruptions in the surrounding area, we should lower the weights of the distances over unreliable features. Thus, we chose the normalization weight $(\alpha_{\vec{i}}^{c^*})^{\gamma}$ for such tuning with the parameter γ. We observed that γ is not a sensitive parameter and that the quality measure results in good separation for a larger range of values. In all our experiments we used $\gamma = 10$. To provide some more intuition for (19), note that the quality measure is proportional to the percentage of the pattern c^* that is being matched in a given patch if the mask parameters are strictly binary, i.e., if a feature is either maximally reliable ($\alpha_{\vec{i}}^{c} = 1$), or maximally unreliable ($\alpha_{\vec{i}}^{c} = 0$). If for instance a patch contains a complete and clean instance of the pattern c^* at position $\vec{x}^*$, $p(m_{\vec{i}} = 1 \,|\, c^*, \vec{x}^*, Y^{(n)}, \Theta)$ is close or equal to one for all reliable features and zero otherwise. This implies that the distance measure is equal to zero and $Q(c^*, \vec{x}^*, Y^{(n)}, \Theta)$ equal to one.

TABLE 2
The Flow of Document Cleaning

```
compute the histogram H_B as the background distribution
learn pattern representations (Tab. 1)
remove non-characters from the learned representations
compute bounding boxes for the character representations
select clean pixel representations for each character representation
repeat
    for each patch of the document do
        compute the best matching pattern class and
        position (c*, x*) using the MAP estimate in (18)
        compute the quality Q of the match using (19)
        if Q ≥ Q_0 and the detected pattern is fully
        inside the patch then
            mark this match as accepted
            paint the clean representation of pattern c*
            at the detected position x*
            erase the reconstructed pattern from the
            original document
        end if
    end for
until no more matches are accepted
```

6 The Document Cleaning Procedure

By making use of the learned character representation, character matching, and match qualities, we can now remove corruptions from a given scanned text document (see the flow of document cleaning in Table 2).

6.1 Preparation

Before going into the details about the document cleaning procedure, some preparations have to be made. Of all learned pattern representations some may, for instance, not represent characters and will have to be identified as such. Furthermore, the cleaning procedure will require a clean representation of each character for reconstruction.

The class number C assumed by the learning algorithm may not be equal to the number of character types in the training data set. A flexible approach is to set C larger than the actual number of character types and to try to identify after learning those representations that do not correspond to characters. In numerical experiments on different types of data, we noticed that non-character representations usually have much lower values for learned mask parameters (see Fig. 5b for an example). On the other hand, for the classes representing true characters, the mask parameters in the center area (describing the character shape) are usually very high (close to 1). Based on these observations, we define the set of salient features for each class c as

$$\mathcal{B}^c = \left\{ \vec{i} \mid \alpha^c_{\vec{i}} > \alpha_o \right\}, \tag{20}$$

where in all of our experiments the threshold $\alpha_o = 0.85$ is used. With the definition of salient features, the classes of true characters can be distinguished from non-character ones by counting the number of salient features. In all of our experiments, we observed a clear separation between these two groups of classes (see Fig. 6b for an example). With a simple threshold, e.g., $\frac{1}{3}$ of the highest salient feature number (the threshold used in our experiments), they can easily be separated. If multiple classes model the same character, these classes are identified according to the similarity of features and masks with shift-invariant.

The next step is to compute a tight bounding box for each character type, i.e., estimating for each represented character a rectangular region that contains the character. The bounding boxes will be used for the cleaning procedure later on. The patch size used in our learning algorithm is always much larger than the actual size of the characters as we want every character in the document to be completely inside at least one patch. One consequence is that the learned mask parameters are not cleanly restricted to the character shape as can be observed by considering the learned representations (see, e.g., Fig. 5b). There are some low value areas at the left and right side of the patch because there is often more than one character inside a patch. Therefore, each representation does contain not only the modeled character but also the average of the characters appearing at the left and right side. To find the region inside of each representation that corresponds to a character, we compute a bounding box around the reliable features (see Fig. 6c for an example).

Finally, the document cleaning is achieved by replacing each detected character of the corrupted document by a clean character. As a fully unsupervised approach, we do not have prior knowledge of the character shapes in the document. The clean character representations have to be found from the corrupted document without any label information. Our model builds its internal representations of characters in terms of Gabor wavelets, which do not generate images of characters directly. To obtain clean character representations for reconstruction, we therefore search the entire corrupted document for the best match of each learned class. More precisely, we determine the best match by the highest quality measure with $\gamma = 0$ and cut out a segment the size of the characters bounding box (Fig. 6c shows some examples). In case of a misclassification, the reconstructed character will be significantly different from the original one.

6.2 Document Cleaning

After the autonomous identification of classes representing characters, their associated bounding boxes, and their cleanest examples in pixel space, we can now clean a document by reconstructing each possibly corrupted character by a clean version (Fig. 8 shows an example). For reconstruction, we screen through the corrupted document patch by patch from upper left to lower right and with patches overlapping by about 50 percent. For each patch we compute the match $(c^*, \vec{x}^*)$ according to (18) and the match quality using (19). If the matched position $\vec{x}^*$ corresponds to a pattern fully visible within the patch, and if the match quality is above the threshold $Q_0 = 0.5$, we paint the best representation of class c^* at position $\vec{x}^*$ onto an initially blank reconstructed document. Fig. 7 illustrates this procedure for a small area of the example document.

As can be observed, not all the matches are accepted for reconstruction because some matches correspond to patterns not entirely visible (e.g., the second patch at iteration 1) or because match qualities are too low (e.g., the last patch at iteration 2). The quality threshold prevents dirt from being reconstructed as characters. As for each patch just one match is computed, not all characters are reconstructed at first. For a complete reconstruction we therefore erase each successfully reconstructed character in the original document by painting a blank rectangle (of the same size as the corresponding bounding box) and apply the procedure again. Patterns that previously were not identified because of competition with other patterns can now be found and correctly reconstructed. The reconstruction procedure terminates once no more matches are accepted.

In Fig. 7 two iterations through the document are sufficient to successfully reconstruct the word "bayes". The entire document in Fig. 8 is perfectly reconstructed after three iterations. The more a document is corrupted by dirt, the less perfect we can expect the reconstruction to be. In examples with dirt fully occluding parts of the document, we do thus obtain many false negative errors. False positive errors are, on the other hand, obtained if, e.g., a random combination of manual line strokes coincides with the feature arrangement of a learned pattern (see Appendix C, available in the online supplemental material). Although error rates for imperfect reconstructions can be decreased by fine tuning the threshold Q_0, we left the parameter unchanged at $Q_0 = 0.5$ for all examples to demonstrate the generality of the approach.

6.3 Quantitative Comparison

To give a quantitative evaluation of our algorithm we computed the recognition rate (the percentage of the characters

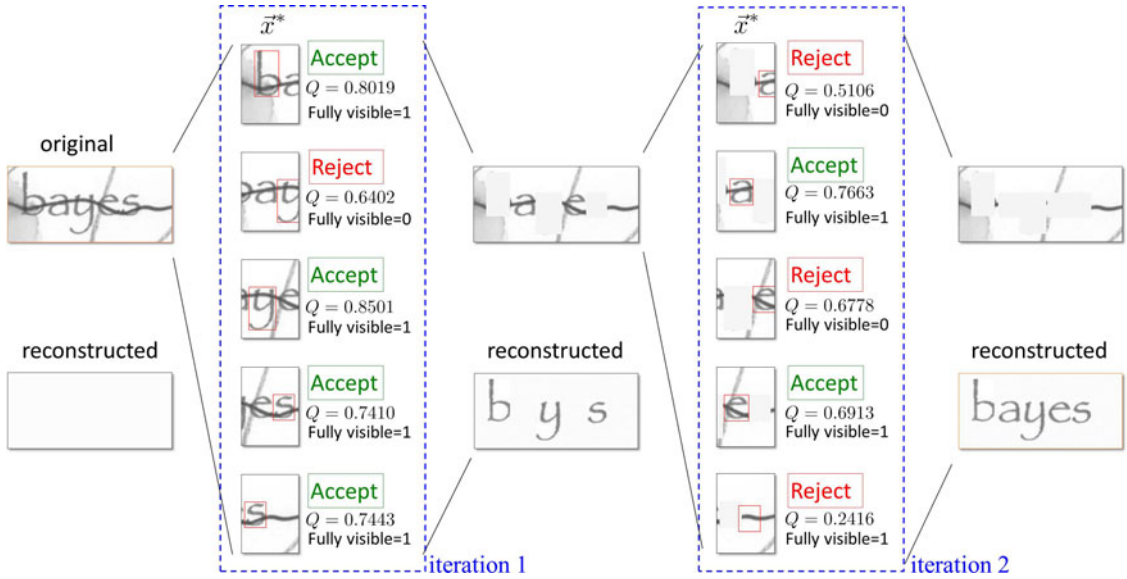
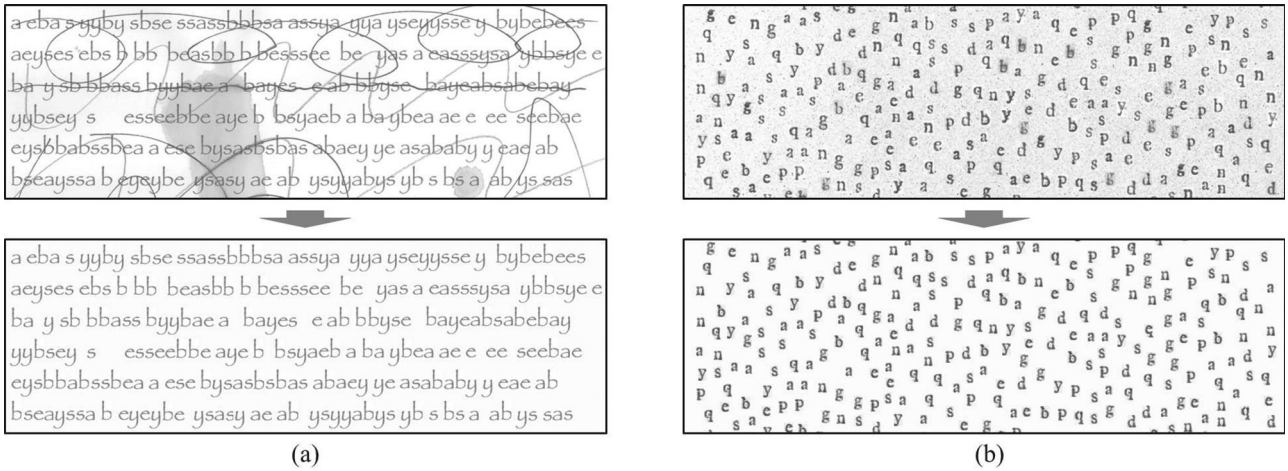


Fig. 7. An illustration of the cleaning procedure. The first column shows the original document (using a small area as an example) and the reconstructed document (initially blank). The second column shows patches of the original document. Using the learned character representations, the MAP estimate of the character class c^* and position $\vec{x}^*$ is visualized for each patch as a red rectangle. For each match the match quality Q is computed and given on the right-hand-side. The match is accepted if the match quality is above a threshold Q_0 and if the matched character is completely inside the patch. The characters of the accepted matches are reconstructed by painting clean characters at the matched positions while the character is erased from the original document (third column). As not all characters can be reconstructed at once, the reconstruction process is iterated until no more characters are accepted for reconstruction (after two iterations for this example).

from the original document being correctly recognized) and the number of false positives (FP) (the number of wrongly recognized characters which do not exist in the original document). Recognition rates and numbers of false positives can be computed for any alphabet, which makes them an appropriate measure for our approach. Quantitative evaluations of other approaches (e.g., [15], [16], [17], [18], [43]) required well-known alphabets (such as the Latin alphabet) because they were based on improvements of OCR before and after the application of the respective image enhancement method. As a baseline comparison, we also applied a state-of-art commercial OCR software (FineReader, [44]) to the same scanned documents used for our approach. The OCR algorithm is only applied to the corrupted documents for comparison because recognition rates (and false positives) for the reconstructed documents would essentially correspond to those of our algorithm (in the case of standard Latin alphabets).

Besides the document shown in Fig. 8a, we made (as briefly mentioned above) several experiments on different types of other documents. For a comparison with conventional approaches, the document cleaning has been performed on a document image consisting of characters of a historical newspaper [42] (see Fig. 8b). Additionally we performed document cleaning on documents with more character types (nine or 12 character types) and on a document



(a) (b)

Fig. 8. Examples of documents cleaned by the described procedure. Top: the corrupted documents. Bottom: the cleaned document. (a) A part of the document used in Section 4.2. (b) A part of the document with characters of an historical newspaper [42]. Five different instances of each character type (in total ten character types) have been used, and the characters were randomly placed on a background that mimicked the texture of historical paper.

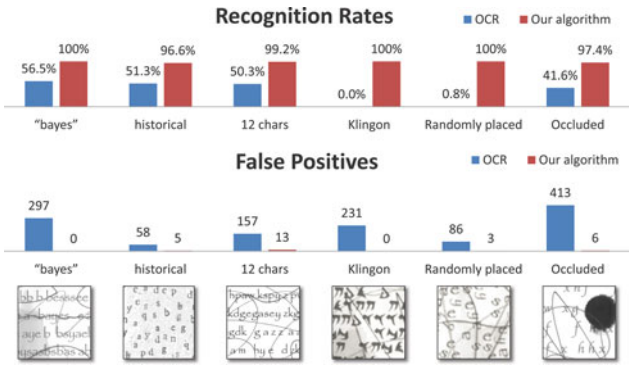


Fig. 9. Quantitative comparison of our algorithm to state-of-art OCR software [44]. Top: The recognition rates (percentages of characters from the original document being correctly recognized) are given for different examples. Bottom: The numbers of false positives (wrongly recognized characters which do not exist in the original document). For each example document we show a small patch in the bottom row. The full documents are shown in the Appendix C, available in the online supplemental material.

with an unusual character set (Klingon was used as an extreme case) to highlight that our approach is not using prior knowledge about character shapes. To highlight that no knowledge about text alignment is used, we furthermore show results on a document with randomly placed and rotated characters. Finally, to show false positives and failed reconstructions of some characters, we performed an experiment on a document containing dark ink spots and ambiguous patterns (compare Appendix C, available in the online supplemental material).

The results of our algorithm and the OCR algorithm are quantitatively evaluated by computing the recognition rate (the percentage of the characters from the original document being correctly recognized) and the number of false positives (the number of wrongly recognized characters which do not exist in the original document).[3] For the document in Fig. 8a, for instance, FineReader recognized 56.5 percent of the characters correctly (essentially those that are segmentable) and corruption by dirt caused 297 false positives. On the same data, our approach detected 100 percent of the characters correctly with no false positives. Fig. 9 shows the results for Fig. 8b and summarizes results for other examples. The poorest performance of FineReader in all the examples is observed for documents with non-standard characters or unusual character orientations. For the documents with Klingon and randomly placed rotated characters, for instance, FineReader resulted in recognition rates of 0 percent (231 FP) and 0.8 percent (86 FP), respectively. For comparison, our approach detected 100 percent (no FP) and 100 percent (3 FP), respectively. Typical cases for false positives and misclassifications of our algorithm are highlighted in Appendix C, available in the online supplemental material, (Figs. 14, 16, 21 and others). One example shows misclassifications caused by high similarities between characters.

3. Note that the numbers of false positives for the OCR algorithm are only rough estimates because its character segmenter often cuts a characters into multiple segments or groups multiple characters into one segment.

For instance, in documents containing "m" and "n" characters, our approach can interpret a patch containing an "m" pattern as a corrupted "n". To further improve performance, cases such as classifications of character subpattern can explicitly be addressed. Other cases such as strong occlusions represent more principled limitations.

The reason behind the poor performance of conventional OCR algorithm is that it first needs to segment characters out based on the statistical information of text alignment, and then recognize those characters based on pre-trained character classifiers. In the problem of document cleaning, the document is corrupted, e.g., with line strokes consisting of the same type of basic features as characters. Such a corruption severely affects the character segmentation processing and poses considerable challenges to the character classifiers, which results in a poor performance in the document cleaning task. Note, however, that a comparison of an OCR algorithm to our approach on these data is not fair. The only reason for the comparison is to provide a baseline performance of the most related approach. OCR software is not intended for the task addressed here, as it is not trained on the corrupted data and as it does not aim for cleaning a document independent of the alphabet. Vice versa, our algorithm would not perform well on typical OCR tasks.

7 Discussion

We have studied an unsupervised approach to clean corrupted scanned documents. Our approach relies on the learning of character representations using a probabilistic generative model with explicit position encoding. Similar to other probabilistic approaches, e.g., image denoising, we followed the general principle of capturing the regularities of the data, and removed unwanted data parts after identifying them as deviations from the learned regularities. However, in contrast to approaches for noise removal, we learned explicit high-level representations of specific image components, i.e., of characters. Having an explicit notion of feature arrangements per character allows for a discrimination of irregular patterns versus characters even though these irregular patterns can consist of the same features (line strokes) as the characters themselves. Methods not representing characters explicitly (e.g., [29]) are, therefore, not applicable or would, at the least, require additional mechanisms to identify characters and to discriminate them against irregular patterns. The idea of using statistical information from patches of corrupted/degraded document in order to improve them has been explored before. Such document patches contains redundant information about the characters of the document and, therefore, can be used to solve tasks at various levels: from denoising the document image, over enhancing/recovering the document image [17], [18], [43], to learning of character representations for their identification and reconstruction (our approach). Our method distinguishes itself from previous approaches in the following aspects: (1) we work with larger patches which can contain multiple characters; (2) we explicitly learn character representations, which provides an explicit separation between meaningful characters and severe corruptions/degradations even if characters and corruptions share many features; (3) our approach can directly identify

characters, which is as powerful as an OCR algorithm without having to be trained on labeled data; (4) we take advantage of sophisticated image features and are robust to small distortions and degradations.

By applying our approach we have shown in this study that even under difficult conditions a perfect reconstruction of a text document is possible with solely the information on a single page. The result of the cleaning procedure depended on factors like the severity of the corruption, the number of character instances per character type, and on the similarity between character patterns and corrupting patterns. Very simple characters like "I", "X", "V" or "C" are, for instance, easier to confuse with random line strokes than more complex characters; and regular line strokes (same orientation and thickness) may be learned as foreground objects. Furthermore, the more character types a document contains the more challenging the discrimination between characters becomes, especially for strongly corrupted data. This is true for learning as well as for character identification. Regarding required data, we usually observed good result in our experiments for more than 200 character instances per character type. Performance significantly decreased for less than 100 instances, primarily due to less appropriate learning of the character representations. The example of Fig. 8 contains about 250 instances per character type ($1{,}251$ characters in total). For the same number of characters, text with 12 character types (about 100 instances per type) could still be processed with low error rates (compare '12chars' example). A similar page with text consisting of the full alphabet of letters, even if constrained to just lower or upper case, would not provide sufficiently many character examples for self-cleaning, however. A natural extension of the addressed task to more character types would, therefore, require several pages. If we assume that about 200 examples per character type are needed and if a page contains $1{,}000$ characters in total, we would require about six pages to learn a full Latin alphabet of lower-case letters. For the general type-set of all letters and numbers (excluding special characters), we would require about 13 pages. If we, furthermore, consider that, e.g., only just 0.074 percent of all characters in the English language are of type 'z' [45], then the number of required pages would increase to about 270. To execute the cleaning procedure described in this work, processing of 270 pages amounts to unreasonably long computation times (even using parallel implementations). The computational effort and the limitation in the size of alphabets is also a clear distinction to above discussed alternative approaches [13], [14], [15], [16], [17], [18]. Because of these limitations, such previously approaches are still clearly preferable for concrete applications including the enhancement of historical documents or the improvement of OCR approaches.

In future work the performance of our approach can be further improved by exploiting further regularities of words and text. The regular arrangement of characters along a line (compare [46]) could be used to predict the positions of characters, and linguistic regularities (e.g., probabilistic language models) could be used to predict character types from context. Using probabilistic generative approaches, such prior knowledge can be integrated into the model by constructing or by learning more sophisticated prior distributions $p(c, \vec{x} \,|\, \Theta)$. Also on the algorithmic side further improvements can be made, e.g., by using a multiple-cause structure (e.g., [47], [48]) to recognize multiple patterns in a patch simultaneously, or by using image features with scale invariance and contrast normalization (e.g., SIFT [49], HOG [50]). Different font sizes of characters can be handled by modeling them as different patterns, by estimating font sizes with heuristic mechanisms, or by adding scaling transformations to the model. An extended set of transformations would, however, further increase the hidden state space and the associated computational demand. The efficiency of the learning algorithm could be further improved by exploiting the techniques in object detection literature. In our E-step, brute-force sliding window computation has been avoided by selecting a small number of candidate translations according to a subset of features. Beyond that, invariant features [9] or techniques like coarse-to-fine search can, in principle, be used for speeding up the selection procedure. Such techniques could dramatically reduce the search space but would imply a coarse-to-fine pyramid structure of character representations, which is much more complicated than our current grid representation. Therefore, it is beyond the scope of this work.

To summarize, by applying the probabilistic approach described in this work, we have shown that it is in principle possible to autonomously clean text documents which are heavily corrupted by irregular patterns. Future developments can further improve the cleaning performance by exploiting regularities of words and sentences, or they can extend the application domain of the approach.

Acknowledgments

This work was funded by the German Research Foundation (DFG) under grant LU 1196/4-2. Large parts of the research under the grant were done at the Frankfurt Institute for Advanced Studies of the Goethe-University Frankfurt, Germany (the previous institution of the authors).

References

[1] U. Schmidt, Q. Gao, and S. Roth, "A generative perspective on MRFs in low-level vision," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2010, pp. 1751–1758.

[2] R. Kindermann and J. L. Snell, *Markov Random Fields and Their Applications*. Providence, RI, USA: Am. Math. Soc., 1980.

[3] S. Z. Li, *Markov Random Field Modeling in Image Analysis*. New York, NY, USA: Springer, 2009.

[4] B. A. Olshausen and D. J. Field, "Emergence of simple-cell receptive field properties by learning a sparse code for natural images," *Nature*, vol. 381, pp. 607–609, 1996.

[5] H. Lee, A. Battle, R. Raina, and A. Y. Ng, "Efficient sparse coding algorithms," in *Proc. Adv. Neural Inf. Process. Syst.*, 2007, pp. 801–808.

[6] J. Wright, Y. Ma, J. Mairal, G. Sapiro, T. Huang, and S. Yan, "Sparse representation for computer vision and pattern recognition," *Proc. IEEE*, vol. 98, no. 6, pp. 1031–1044, Jun. 2010.

[7] B. A. Olshausen, C. H. Anderson, and D. C. V. Essen, "A neurobiological model of visual attention and invariant pattern recognition based on dynamic routing of information," *The J. Neurosci.*, vol. 13, no. 11, pp. 4700–4719, 1993.

[8] L. Wiskott, J.-M. Fellous, N. Krüger, and C. von der Malsburg, "Face recognition by elastic bunch graph matching," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, no. 7, pp. 775–779, Jul. 1997.

[9] B. J. Frey and N. Jojic, "Transformation-invariant clustering using the EM algorithm," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, pp. 1–17, Jan. 2003.
[10] D. B. Grimes and R. P. N. Rao, "Bilinear sparse coding for invariant vision," *Neural Comput.*, vol. 17, pp. 47–73, 2005.
[11] C. K. I. Williams and M. K. Titsias, "Greedy learning of multiple objects in images using robust statistics and factorial learning," *Neural Comput.*, vol. 16, pp. 1039–1062, 2004.
[12] J. Mantas, "An overview of character recognition methodologies," *Pattern Recognit.*, vol. 19, no. 6, pp. 425–430, 1986.
[13] G. Kopec and P. Chou, "Document image decoding using Markov source models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 16, no. 6, pp. 602–617, Jun. 1994.
[14] G. E. Kopec and M. Lomelin, "Document-Specific Character Template Estimation," in *Proc. SPIE*, vol. 2660, 1996, pp. 14−26.
[15] M. Bern and D. Goldberg, "Model-based document image improvement," in *Proc. Int. Conf. Image Process.*, 2000, pp. 582–585.
[16] Q. Zheng and T. Kanungo, "Morphological Degradation Models and their Use in Document Image Restoration," in *Proc. Int. Conf. Image Process.*, 2001, pp. 193–196.
[17] R. F. Moghaddam and M. Cheriet, "Beyond pixels and regions: A non-local patch means (NLPM) method for content-level restoration, enhancement, and reconstruction of degraded document images," *Pattern Recognit.*, vol. 44, no. 2, pp. 363–374, 2011.
[18] J. Banerjee, A. Namboodiri, and C. Jawahar, "Contextual restoration of severely degraded document images," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2009, pp. 517–524.
[19] M. Titsias and C. Williams, "Fast unsupervised greedy learning of multiple objects and parts from video," in *Proc. Conf. Comput. Vis. Pattern Recognit. Workshop*, 2004, p. 179.
[20] N. Jojic and B. J. Frey, "Learning flexible sprites in video layers," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2001, pp. 517–524.
[21] B. J. Frey, N. Jojic, and A. Kannan, "Learning appearance and transparency manifolds of occluded objects in layers," in *Proc. IEEE CS Conf. Comput. Vis. Pattern Recognit.*, 2003, pp. 45–52.
[22] A. Kannan, N. Jojic, and B. J. Frey, "Generative model for layers of appearance and deformation," in *Proc. 10th Int. Workshop Artif. Intell. Statist.*, 2005, pp. 166–173.
[23] J. Winn and A. Blake, "Generative affine localisation and tracking," in *Proc. Adv. Neural Inf. Process. Syst.*, 2004, pp. 1505–1512.
[24] A. Kannan, N. Jojic, and B. J. Frey, "Fast Transformation-Invariant Component Analysis," *Int. J. Comput. Vis.*, vol. 77, no. 1-3, pp. 87–101, 2007.
[25] J. D. Jackson, A. J. Yezzi, and S. Soatto, "Dynamic shape and appearance modeling via moving and deforming layers," *Int. J. Comput. Vis.*, vol. 79, no. 1, pp. 71–84, 2008.
[26] C. Wang, M. D. L. Gorce, and N. Paragios, "Segmentation, ordering and multi-object tracking using graphical models," in *IEEE 12th Int. Conf. Comput. Vis.*, 2009, pp. 747–754.
[27] J. Y. A. Wang and E. H. Adelson, "Representing moving images with layers," *IEEE Trans. Image Process.*, vol. 3, no. 5, pp. 625–38, Sep. 1994.
[28] D. Sun, E. B. Sudderth, and M. J. Black, "Layered segmentation and optical flow estimation over time," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2012, pp. 1768–1775.
[29] N. Jojic, B. Frey, and A. Kannan, "Epitomic analysis of appearance and shape," in *Proc. 9th IEEE Int. Conf. Comput. Vis.*, 2003, pp. 34–41.
[30] K. Ni, A. Kannan, A. Criminisi, and J. Winn, "Epitomic location recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 12, pp. 2158–67, Dec. 2009.
[31] J. Lücke and J. Eggert, "Expectation truncation and the benefits of preselection in training generative models," *The J. Mach. Learn. Res.*, vol. 11, pp. 2855–2900, 2010.
[32] R. M. Neal and G. E. Hinton, "A view of the EM algorithm that justifies incremental, sparse, and other variants," in *Proc. NATO Adv. Study Inst. Learn. Graph. Models*, 1998, pp. 355–368.
[33] M. Jordan, Z. Ghahramani, T. Jaakkola, and L. Saul, "An introduction to variational methods for graphical models," *Mach. Learn.*, vol. 37, pp. 183–233, 1999.
[34] E. Körner, M. O. Gewaltig, U. Körner, A. Richter, and T. Rodemann, "A model of computation in neocortical architecture," *Neural Netw.*, vol. 12, pp. 989–1005, 1999.
[35] V. A. F. Lamme and P. R. Roelfsema, "The distinct modes of vision offered by feedforward and recurrent processing," *Trends Neurosci.*, vol. 23, no. 11, pp. 571–579, 2000.
[36] R. D. S. Raizada and S. Grossberg, "Towards a theory of the laminar architecture of cerebral cortex: Computational clues from the visual system," *Cerebral Cortex*, vol. 13, pp. 100–13, 2003.
[37] A. Yuille and D. Kersten, "Vision as Bayesian inference: Analysis by synthesis?" *Trends Cognitive Sci.*, vol. 10, no. 7, pp. 301–8, 2006.
[38] S. Madec, A. Rey, S. Dufau, M. Klein, and J. Grainger, "The time course of visual letter perception," *J. Cognitive Neurosci.*, vol. 24, no. 7, pp. 1645–1655, 2012.
[39] G. Puertas, J. Bornschein, and J. Lücke, "The maximal causes of natural scenes are edge filters," in *Proc. Adv. Neural Inf. Process. Syst.*, 2010, vol. 23, pp. 1939–1947.
[40] L. Shen and L. Bai, "A review on Gabor wavelets for face recognition," *Pattern Anal. Appl.*, vol. 9, pp. 273–292, 2006.
[41] Y. LeCun, K. Kavukcuoglu, and C. Farabet, "Convolutional networks and applications in vision," in *Proc. IEEE Int. Symp. Circuits Syst.*, 2010, pp. 253–256.
[42] "Les gazettes europeennes du 18eme siecle," [Online]. Available: http://gazettes18e.ish-lyon.cnrs.fr/
[43] L. Likforman-Sulem, J. Darbon, and E. H. B. Smith, "Enhancement of historical printed document images by combining total variation regularization and non-local means filtering," *Image Vis. Comput.*, vol. 29, no. 5, pp. 351–363, 2011.
[44] "Abbyy finereader 11," (2011). [Online]. Available: http://finereader.abbyy.com/
[45] H. Beker and F. Piper, *Cipher Systems: The Protection of Communications*. Hoboken, NJ, USA: Wiley-Interscience, 1982.
[46] R. G. Casey and E. Lecolinet, "A survey of methods and strategies in character segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 18, no. 7, pp. 690–706, Jul. 1996.
[47] P. Dayan and R. S. Zemel, "Competition and multiple cause models," *Neural Comput.*, vol. 7, pp. 565–579, 1995.
[48] Z. Dai and J. Lücke, "Unsupervised learning of translation invariant occlusive components," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2012, pp. 2400–2407.
[49] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, pp. 91–110, 2004.
[50] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2005, pp. 886–893.

Zhenwen Dai received the BSc degree in computer science from Zhejiang University, China, in 2007, and the MPhil degree in computer science from The University of Hong Kong, in 2009, and the doctoral degree in computer science from the Goethe-University Frankfurt, in 2013. He is currently a postdoctoral research associate at the University of Sheffield in the field of machine learning and computer vision. He is a member of the IEEE.

Jörg Lücke received the PhD degree from the Ruhr-University Bochum, Germany, in 2005 and then joined the Gatsby Computational Neuroscience Unit, UCL, United Kingdom, as a postdoc. With grants from different funding agencies, he then built up his own research group at the Frankfurt Institute for Advanced Studies, Goethe-University Frankfurt, and later at the Technical University Berlin. Since 2013, he is an associate professor of machine learning at the University of Oldenburg, Germany. He is a member of the IEEE.