Scene Text Recognition using Similarity and a Lexicon with Sparse Belief Propagation

Jerod J. Weinman, *Member, IEEE*, Erik Learned-Miller, *Member, IEEE*, Allen R. Hanson *Member, IEEE*

Abstract—Scene text recognition (STR) is the recognition of text anywhere in the environment, such as signs and store fronts. Relative to document recognition, it is challenging because of font variability, minimal language context, and uncontrolled conditions. Much information available to solve this problem is frequently ignored or used sequentially. Similarity between character images is often overlooked as useful information. Because of language priors, a recognizer may assign different labels to identical characters. Directly comparing characters to *each other*, rather than only a model, helps ensure that similar instances receive the same label. Lexicons improve recognition accuracy but are used *post hoc*. We introduce a probabilistic model for STR that integrates similarity, language properties, and lexical decision. Inference is accelerated with sparse belief propagation, a bottom-up method for shortening messages by reducing the dependency between weakly supported hypotheses. By fusing information sources in one model, we eliminate unrecoverable errors that result from sequential processing, improving accuracy. In experimental results recognizing text from images of signs in outdoor scenes, incorporating similarity reduces character recognition error by 19%, the lexicon reduces word recognition error by 35%, and sparse belief propagation reduces the lexicon words considered by 99.9% with a 12X speedup and no loss in accuracy.

Index Terms—Scene text recognition, optical character recognition, conditional random fields, factor graphs, graphical models, lexicon, language model, similarity, belief propagation, sparse belief propagation

1 Introduction

The problem of optical character recognition (OCR), or the recognition of text in machine-printed documents, has a long history and is one of the most successful applications of computer vision, image processing, and machine learning techniques. In this paper, we focus on scene text recognition (STR), the recognition of text anywhere in the environment, such as on store fronts, traffic signs, movie marquees, or parade banners. While superficially similar to OCR, STR is significantly more challenging because of extreme font variability, uncontrolled viewing conditions, and minimal language context. Difficult viewing angles, shadows, occlusions, unique fonts, and lack of language context are all problems that make the typical STR problem signficantly more difficult than a straightforward OCR application.

In fact, while state-of-the-art OCR systems typically achieve character recognition rates over 99% on clean documents, they fail catastrophically on STR problems. Humans of course, have no trouble reading text in the environment under normal conditions. One reason for the gap between human and machine performance in STR problems could be that people are able to apply many more sources of information to the problem than current automated techniques. This is not unique to character recognition, of course; using more information sources in approaches to many computer vision problems should improve results. Because several sources of information factor into the reading process, we require a computational model that can combine factors in a unified framework.

In this paper, we propose a probabilistic graphical model for STR that brings together bottom-up and top-down information as well local and long-distance relationships into a single elegant framework. In addition to individual character *appearance*, our model integrates *appearance similarity*, one underused source of information, with local language statistics and a lexicon in a unified probabilistic framework to reduce false matches—errors in which the different characters are given the same label—by a factor of four and improve overall accuracy by greatly reducing word error. The model adapts to the data present in a small sample of text, as typically encountered when reading signs, while also using higher level knowledge to increase robustness.

The paper is organized as follows. In the remainder of this section, we give additional background on using similarity and lexicons in text recognition and discuss why sparse belief propagation is important. The next section briefly introduces the discriminative probabilistic framework in terms of factor graphs. Section 3 describes the particular form and features of our model for scene text recognition. We describe the particulars and advantages of sparse belief propagation for efficient approximate inference in Section 4. Our experimental results on scene text images are presented in Section 5, and we conclude in Section 6.

1.1 Similarity

Because of language priors or other constraints, it is not unusual for a text recognizer to assign two different labels to two identical characters (see Figure 1). This is particularly common when the appearance model for a font is weak, causing high character ambiguity. This happens frequently in STR due to the lack of training data for a particular font.

- *Jerod Weinman is with the Department of Computer Science at Grinnell College. Email: weinman@grinnell.edu.*
- *Erik Learned-Miller and Allen Hanson are with the Department of Computer Science at the University of Massachusetts Amherst. Email: {elm,hanson}@cs.umass.edu.*

Portions of this manuscript are reprinted, with permission, from papers appearing in the 2006 IEEE Conference on Computer Vision and Pattern Recognition [1], ©2006 IEEE, and 2007 IAPR International Conference on Document Analysis and Recognition [2], ©2007 IEEE.

Information	Result
Appearance	`FIeet`
Appearance, Language	`Fteat`
Appearance, Language, Similarity	`Fleet`
Similarity → Appearance, Language	`Fteet`

Fig. 1. A query image (top) is interpreted with varying amounts of image and linguistic information. Only when unified with similarity information is the other contextual information constrained to global consistency.

In an effort to make character recognition more robust to font variations or noise, recent advances in OCR performance have exploited the length of documents to leverage multiple appearances of the same character. Hong and Hull [3] cluster word images and then label the clusters, rather than individual words. Similarly, Breuel learns a probability of whether two images contain the same character and uses the probability to cluster individual characters [4], with subsequent cluster labeling (i.e., by voting) and nearest neighbor (most similar) classification [5]. These methods capitalize on the idea of similarity, that characters and words of similar appearance should be given the same label. However, they suffer from the drawback that there is no feedback between the labeling and clustering process. Hobby and Ho [6] ameliorate this somewhat by purging outliers from a cluster and matching them to other clusters where possible. These processes all solve the clustering and recognition problems in separate stages, making it impossible to recover from errors in the clustering stage. Furthermore, they rely on having hundreds or thousands of characters in each recognition problem so that clustering is practical, making them impractical for reading short text segments like those encountered in the STR problem.

Prior to our recent work [1], the *dissimilarity* between character images had not been used as evidence *against* giving them the same label, but in many circumstances this too is a reasonable approach. Previous clustering-based methods only ensure that all cluster members are given the same label; they do not prevent different clusters from being assigned the same label.

Consider the example in Figure 1. The top row of text is the result of reading the sign using only basic information about character images, and the lowercase `l` (*ell*) is mistaken for an uppercase `I` (*eye*). The next result combines the image information with some basic local language information. This does not correct the error but in fact introduces new errors. The image and language information is based on local context and do not require any global consistency. By factoring in character similarity information in the third line, the errors are corrected; the two `e` characters that appear the same are given the same label, while the `l` and `t` characters of dissimilar appearance are given different labels. In contrast, using similarity information first to determine which characters are the same and then identifying character clusters, as shown in the last line, does not perform as well as a unified model. This is particularly true when the number of characters is very small. We present in the first part of this paper a model that incorporates all of these important information sources [1].

Kumar and Hebert [7] have used such a strategy for the general image labeling problem, associating image sites with particular labels and biasing the eventual classification by measuring the similarity of neighboring image regions. Our approach broadens this to incorporate all pairs of characters that must be classified, not just neighboring characters.

1.2 Lexicon

Higher-level information, such as a lexicon, can also help improve recognition. When images have low-resolution or contain uncommon fonts, a lexicon can inform an otherwise unreliable character recognizer. Humans are more reliable at reading characters when they are present in words, or pseudo-words [8], motivating us to consider how this information may be incorporated in the recognition process. The earliest unification of character confusion likelihoods with a lexicon constraint is by Bledsoe and Browning [9]. Device-specific character confusion likelihoods are combined with word unigram probabilities to find the most likely dictionary word given the OCR output. One major drawback is that the computational load is linear in the size of the lexicon. For low-resolution document images, Jacobs et al. [10] have improved accuracy by forcing character parses to form words drawn from a lexicon, but this technique will not be able to correctly recognize non-lexicon words. In the STR problem, non-words are quite common due to the abundance of proper names.

Related work on specialized models for scene text recognition either ignores helpful contextual and lexical information or incorporates them in an *ad hoc* fashion. For instance, after isolated character recognition, Thillou et al. [11] post-process results by applying an n-gram model to the n-best list of characters. Beaufort and and Mancas-Thillou [12] similarly use a lexicon in a post-processing stage with a finite state machine. Linguistic processing is divorced from recognition in both cases by ignoring the relative probability of characters based on their appearance. Alternatively, Zhang and Chang [13] have handled this by explicitly including a lexical decision variable in a probabilistic model. However, their model does not include local language properties, such as bigrams, for the case when a word is not in the lexicon. We find both are important for recognition accuracy.

As indicated above, one practical issue with using a lexicon is the time it takes to examine candidate words in a large lexicon. Lucas [14] addresses this issue by reusing computation in a trie-formatted lexicon. Another approach, taken by Schambach [15], is to eliminate words from consideration based on the low probability of their constituent characters.

We present an addition to our discriminative model that incorporates a bias for strings from a lexicon [2]. Modeling the lexical decision process allows word predictions to come from outside the lexicon, based on the evidence and a prior bias for (or against) lexicon words. Our model allows efficient approximate inference schemes by eliminating the need to explicitly consider all possible strings, only evaluating entries from the lexicon. Notably, we can speed this process even further by applying an approximate “sparse inference” technique.

1.3 Sparse Belief Propagation

Belief propagation (BP) is a popular and simple method for learning and prediction in probabilistic graphical models. The algorithm's message passing operations require sums over the domain of functions (factors) measuring local compatibility between assigned labels. Since the complexity of the sum grows exponentially with the number of arguments to these functions, it is typically only used with factors of two or three unknowns. Adding a lexicon to our model introduces factors over several more unknowns—the characters of an entire word. Without simplification and approximation, using belief propagation would be untenable. Fortunately, non-lexicon information allows us to make reasonable predictions about which characters are not possible candidates. We use the sparse belief propagation algorithm proposed by Pal et al. [16] toward this end. By eliminating unlikely characters from consideration in messages passed between nodes of a graphical model, we can drastically reduce the set of words that must be considered. Previous work by Coughlan and Shen [17] features an approach similar in spirit for pairwise functions, but it has not been generalized to higher-order functions, and it uses thresholds for sparsity that may not provide good approximations to the messages. Importantly, sparse belief propagation does *not* completely drop characters from consideration. Rather, it merely reduces the contextual dependence between characters when one of them has weak support based on other information. Further details are given in Section 4.

In summary, by fusing the available information sources, such as character similarity and a lexicon, in a single model, we improve overall accuracy and eliminate unrecoverable errors that result from processing the information in separate stages.

2 Probabilistic Framework

Graphical models of probability are a powerful tool for describing and modeling the logical dependence of various information sources and unknowns in a Bayesian framework. We employ a discriminative undirected graphical model [18] for predicting character identities.

Let $\mathbf{x}$ be an input image representation and $\mathbf{y} = y_1 y_2 \ldots y_n$ be the string of characters contained in the image, taken from an alphabet Y. Letting I represent our information and assumptions about the problem, we frame the task of reading text in images as an inference problem—using I and some training data $\mathcal{D}$—over a model or parameter space Θ:

$$p(\mathbf{y} \mid \mathbf{x}, \mathcal{D}, I) = \int_{\boldsymbol{\Theta}} p(\mathbf{y} \mid \mathbf{x}, \boldsymbol{\theta}, I)\, p(\boldsymbol{\theta} \mid \mathcal{D}, I)\, \mathrm{d}\boldsymbol{\theta}. \tag{1}$$

Note we have assumed that (i) given a prediction model $\boldsymbol{\theta}$, the training data $\mathcal{D}$ do not reveal anything additional about $\mathbf{y}$, and (ii) given the training data $\mathcal{D}$, an additional image $\mathbf{x}$ does not give any information about the prediction model $\boldsymbol{\theta}$. Evaluating such an integral is non-trivial, so we take the standard approach of finding the most likely model

$$\widehat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} p(\boldsymbol{\theta} \mid \mathcal{D}, I) \tag{2}$$

and using the point approximation $p(\boldsymbol{\theta} \mid \mathcal{D}, I) = \delta\left(\boldsymbol{\theta} - \widehat{\boldsymbol{\theta}}\right)$ so that the integral (1) becomes

$$p(\mathbf{y} \mid \mathbf{x}, \mathcal{D}, I) \approx p\left(\mathbf{y} \mid \mathbf{x}, \widehat{\boldsymbol{\theta}}, I\right). \tag{3}$$

The probability $p(\mathbf{y} \mid \mathbf{x}, \boldsymbol{\theta}, I)$ is the typical undirected graphical model. Let C contain some subset of the $\{1, \ldots, n\}$ positions of $\mathbf{y}$, so that $\mathbf{y}_C$ gives the values of the subset. The conditional probability is expressed as a product of local factors,

$$p(\mathbf{y} \mid \mathbf{x}, \boldsymbol{\theta}, I) = \frac{1}{Z(\mathbf{x}, \boldsymbol{\theta})} \prod_{C \in \mathcal{C}} f_C(\mathbf{y}_C, \mathbf{x}), \tag{4}$$

where $Z(\mathbf{x}, \boldsymbol{\theta})$ is the observation-dependent normalizing factor ensuring the expression is a proper probability distribution. The non-negative factors f_C express the local compatibility among the unknowns in $\mathbf{y}_C$ and the observation $\mathbf{x}$, and $\mathcal{C}$ is a collection of the subsets for indexing these factors. Typically there are several categories of factors that are instantiated several times in the product (4). Each of these instantiations involves the same function, but accepts a different set of arguments C. For example, the same character recognition function is applied at many locations in the image.

2.1 Inference

The index sets C induce a bipartite graph between the factors f_C and the unknowns $\mathbf{y}$, as illustrated in Figure 2. When this graph (not including $\mathbf{x}$ and edges connected to it) is a tree, exact inference may be performed efficiently via the sum-product algorithm [19], also known as belief propagation (BP). Local information stored in the factors influences the global interpretation by passing messages between the factors and nodes. Factors neighboring node i in the graph are indexed by members of the set $\mathcal{N}(i) = \{C \in \mathcal{C} \mid i \in C\}$. The node-to-factor messages have the form

$$m_{i \to C}(y_i) \quad \propto \quad \prod_{C' \in \mathcal{N}(i) \setminus C} m_{C' \to i}(y_i), \tag{5}$$

the product of all the incoming messages to a node from other neighboring factors. The resulting functional message is normalized (sums to 1 over y_i) for numerical stability. The factor-to-node messages combine the local information expressed in the factor and the current messages from its other arguments,

$$m_{C \to i}(y_i) = \sum_{\mathbf{y}_{C \setminus \{i\}}} f_C(\mathbf{y}_C, \mathbf{x}) \prod_{j \in C \setminus \{i\}} m_{j \to C}(y_j). \tag{6}$$

Note that the summation is taken over all values of the nodes in the set $C \setminus \{i\}$.

If the graph has cycles, these messages are iteratively passed until convergence, which is not guaranteed but empirically tends to give reasonable results in many applications. Greater detail about factor graphs and inference may be found in an article by Kschischang et al. [19]. We add more about a sparse version of BP for accelerating inference in Section 4.

2.2 Training

To learn from training data, the probability distribution (4) is parameterized by $\boldsymbol{\theta}$, with each factor f_C having some subset of the parameters, $\boldsymbol{\theta}_C$, as arguments. Given a set of independent, labeled examples $\mathcal{D} = \left\{\mathbf{y}^{(k)}, \mathbf{x}^{(k)}\right\}_k$, the parameters may be estimated by maximizing the posterior probability $p(\boldsymbol{\theta} \mid \mathcal{D}, I)$. The optimization (2) is the usual maximum *a posteriori* (MAP) estimation with some parameter prior $p(\boldsymbol{\theta} \mid I)$ [18]. Using Bayes' rule, the parameter posterior is

$$p(\boldsymbol{\theta} \mid \mathcal{D}, I) \propto p(\boldsymbol{\theta} \mid I) \prod_k p\left(\mathbf{y}^{(k)} \mid \mathbf{x}^{(k)}, \boldsymbol{\theta}, I\right) \tag{7}$$

where the product terms have the same model form (4). After taking logarithms, the objective function is given by

$$\begin{aligned} \mathcal{L}(\boldsymbol{\theta}; \mathcal{D}) = \log p(\boldsymbol{\theta} \mid \boldsymbol{\alpha}, I) + \\ \textstyle\sum_k \sum_{C \in \mathcal{C}^{(k)}} \log f_C\left(\mathbf{y}_C^{(k)}, \mathbf{x}^{(k)}; \boldsymbol{\theta}_C\right) - \log Z\left(\mathbf{x}^{(k)}, \boldsymbol{\theta}\right), \end{aligned} \tag{8}$$

where $p(\boldsymbol{\theta} \mid \boldsymbol{\alpha}, I)$ is a prior on the parameters with conditioning parameters $\boldsymbol{\alpha}$ and information I. The set of factors $\mathcal{C}$ depends on how many characters there are in the observation and is thus indexed by the particular example k. When f_C is log-linear in $\boldsymbol{\theta}_C$ (as all of the learned factors we employ are) and the log prior is convex, the objective $\mathcal{L}(\boldsymbol{\theta}; \mathcal{D})$ is convex, and a global optimum can easily be found.

Because it is a combinatorial sum, the normalization term $Z(\mathbf{x}, \boldsymbol{\theta})$ is generally intractable. To simplify training, we use a piecewise approximation [20], which changes Z from a sum over all $\mathbf{y}$ to a product of local sums over the terms for each factor. Thus, the $\log Z$ term in (8) is replaced by the upper bound $\sum_{C \in \mathcal{C}} \log Z_C$ where

$$Z_C(\mathbf{x}, \boldsymbol{\theta}_C) = \sum_{\mathbf{y}_C} f_C(\mathbf{y}_C, \mathbf{x}; \boldsymbol{\theta}_C). \tag{9}$$

Since the factors are local and typically include only a small set of unknowns, sums over the set of the values for $\mathbf{y}_C$ are practical to compute. Replacing $\log Z$ with an upper bound means we are optimizing a tractable lower bound on the log posterior probability $\mathcal{L}(\boldsymbol{\theta}; \mathcal{D})$.

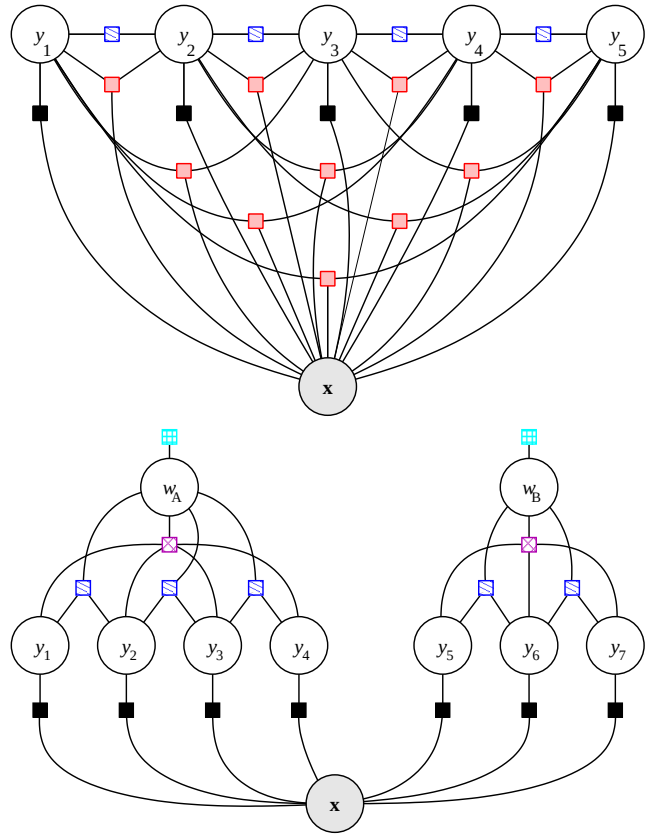


Fig. 2. Factor graphs for inferring characters $\mathbf{y}$ from a given image $\mathbf{x}$. The solid (black) factors capture relationships between the image and character identity (I^A). Hatched (blue) factors between neighboring ys capture language information including bigrams, (I^B), and letter case (I^C). Shaded (red) factors among ys account for similarities between characters in $\mathbf{x}$ for jointly labeling the string (I^S). Cross-hatched (magenta) factors can constrain portions of $\mathbf{y}$ to be drawn from a lexicon, (I^W), while the tiled (cyan) factors capture the bias for lexicon words, (I^L). TOP: Model using pairwise similarity comparisons. BOTTOM: Model incorporating a lexicon and lexical decision unknowns w_A and w_B for two words.

3 Markov Models for Recognition

Using the probabilistic framework described in the previous section involves defining parameterized factors for the data and the labels. For this recognition problem, model input will be size-normalized character images and the output is the predicted character labels. In this section we will outline the details of our model, including the form of the input and features, the relevant information being used, and the particular factors that are learned to form the model.

Our model makes the following assumptions:

1) For each sign, the input is all of the same font
2) Characters have been segmented (that is, the coordinates of their bounding boxes are known), but not binarized
3) Word boundaries are known

Our conditioning information I consists of these in addition to our other basic information. Assumption 1 is especially reasonable for signs containing small amounts of text. Although exceptions to this certainly exist, our database of signs has only a few that stretch the assumption, and it is not difficult to imagine introducing a factor for deciding whether two characters or words are in the same font. While Assumption 2 is not the most general, with high-resolution digital cameras, adequately lit scenes and an area of interest that occupies sufficient area on the sensor, it is reasonable. Note that it does not require a binary image, only a rough localization of each character. Furthermore, with an automated version of Niblack's binarization algorithm [21], we can accurately segment over 96% of characters in our evaluation. Finally, assumption 3 is not overly restrictive since word boundaries can mostly be found by modeling intra- and inter-word character spacings. These assumptions are all reasonable for the problem we are trying to solve, namely, reading short amounts of text found on signs in images of scenes.

In the remainder of this section we build up our model

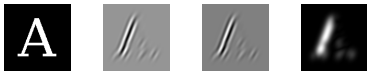


Fig. 3. An example training character with (left to right) real, imaginary, and complex modulus filter responses for one orientation and scale.

from its constituent factors, based on the assumptions listed above. These will reflect several useful sources of information, namely:

- character appearance (*what do As, Bs, etc., tend to look like?*)
- local language properties (*what letters tend to follow other letters? where do we expect capital letters?*)
- character similarity (*which characters look similar or different?*)
- lexicon (*is this string more likely to be elm or clm)?*

Each of these are combined effortlessly into a unified model for character recognition with the basic form outlined in Section 2.

Eventually, we will want to compare the results of the model with various information sources (mathematically represented as factors) included. Since we denote the assumptions or information I that a particular model $p\left(\mathbf{y} \mid \mathbf{x}, \widehat{\boldsymbol{\theta}}, I\right)$ employs, this is used to also indicate the information sources being used. To this end, when a particular class of factors is used, e.g., f^A for character appearance, we indicate this by conditioning the model on the corresponding "information" I^A.

The models resulting from the combination of the factors we will define are shown in Figure 2. All of the various factor types may be combined in one model, but we show them in two separate graphs for clarity. The top graph highlights the "adaptive" model that uses similarity between the character images as part of the recognition process. The bottom graph demonstrates how other factors may be introduced to promote the recognition of strings as lexicon words. Details of each of these factor types are given in the remainder of this section.

3.1 Appearance Model

The most obvious component of a recognition model involves relating character appearances to their identity. Gabor filters are an effective and widely used tool for feature extraction that decompose geometry into local orientation and scale [22]. Their success in handwriting recognition [23] and printed character recognition [24] demonstrates their utility for this task. Using a minimally redundant design strategy [25], a bank of 18 Gabor filters spanning three scales (three full octaves) and six orientations (30° increments from 0° to 150°) is applied to the grayscale image $\mathbf{x}$, yielding complex coefficients $\mathbf{f}$ that contain phase information. The real and imaginary parts of the filter are even and odd functions, respectively.

Taking the complex modulus of the filter outputs $|\mathbf{f}|$ provides phase invariance and makes the responses less sensitive to translations of the input; see Figure 3. Practically, this makes the filter responses invariant to the *polarity* of the text (white-on-black versus black-on-white). After filtering, the complex modulus of each response image is downsized by applying a Gaussian blur and downsampling. This adds a slight amount of insensitivity to feature location for different fonts but mostly serves to reduce the size of the feature vector used as input to the model. All of the downsized responses are collected into a single feature vector for each character, F_i, a function of the original image $\mathbf{x}$.

Given a relationship between the identity of the character and the filter responses, this information is denoted I^A because it is based on the appearance of the character image. We then associate character classes with these filtered images by a log-linear factor

$$\log f_i^A\left(y_i, \mathbf{x}; \boldsymbol{\theta}^A\right) = \boldsymbol{\theta}^A\left(y_i\right) \cdot F_i\left(\mathbf{x}\right). \tag{10}$$

The same appearance parameters are used for every character, so there is no dependence of $\boldsymbol{\theta}^A$ on index i.

3.2 Language Model

Properties of the language are strong cues for recognizing characters in previously unseen fonts and under adverse conditions; much previous work has made use of it in various ways (see, e.g., [26]). We add simple linguistic properties to the model in the form of two information sources: character bigrams and letter case.

It is well known that the English lexicon employs certain character juxtapositions more often than others. N-grams are a widely-used general feature for character and handwriting recognition. Our model uses this information I^B via the log-linear factors

$$\log f_{ij}^B\left(y_i, y_j; \boldsymbol{\theta}^B\right) = \boldsymbol{\theta}^B\left(y_i, y_j\right), \tag{11}$$

where i and j are ordered, adjacent characters within a word. In this model, we do not distinguish letter case in the bigrams, so the weights in $\boldsymbol{\theta}^B$ are tied across case (i.e., $\boldsymbol{\theta}^B(\mathtt{R},\mathtt{A}) = \boldsymbol{\theta}^B(\mathtt{r},\mathtt{A}) = \boldsymbol{\theta}^B(\mathtt{R},\mathtt{a}) = \boldsymbol{\theta}^B(\mathtt{r},\mathtt{a})$).

Prior knowledge of letter case with respect to words also proves important for accurate recognition in English. In some fonts, potentially confusable characters may have different cases (e.g., `l` and `I`, lowercase *ell* and uppercase *eye*, respectively). We can improve recognition accuracy in context because English rarely switches case in the middle of the word. Additionally, uppercase to lowercase transitions are common at the beginning of words, but the reverse is not. Note that digit characters have no case. This information I^C is incorporated with the feature weights

$$\log f_{ij}^C\left(y_i, y_j; \boldsymbol{\theta}^C\right) = \begin{cases} \theta^{C,s} & y_i, y_j \text{ same case} \\ \theta^{C,d} & y_i, y_j \text{ different case} \\ 0 & \text{otherwise,} \end{cases} \tag{12}$$

when i and j are adjacent characters *within* a word and

$$\log f_{ij}^C\left(y_i, y_j; \boldsymbol{\theta}^C\right) = \begin{cases} \theta^{C,u} & y_i \text{ lowercase, } y_j \text{ uppercase} \\ 0 & \text{otherwise,} \end{cases} \tag{13}$$

when i and j are the first and second characters of a word, respectively. Thus, for this letter case model, f^C, we have the parameters $\boldsymbol{\theta}^C = \begin{bmatrix} \theta^{C,s} & \theta^{C,d} & \theta^{C,u} \end{bmatrix}$. Note that the functions (12) and (13) have the same general log-linear form as (10) and (11), but we present their more compact, tied form here.

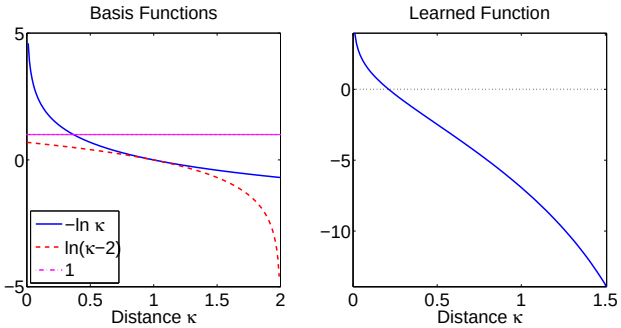


Fig. 4. Similarity basis functions and the learned compatibility for the distance between different images of the same character; the coefficients are $\boldsymbol{\theta}^S = \begin{bmatrix} 0.9728 & 9.3191 & -6.9280 \end{bmatrix}$. The dotted line in the right-hand figure shows the crossover from reward to penalty, which occurs at a vector angle of about 37°.

3.3 Similarity Model

An important, underused source of information for recognition is the similarity among the character images themselves—two character images that look the same should rarely be given different labels. Toward this end, we need a comparison function for images. We have found the vector angle between the concatenated real and imaginary parts

$$\mathbf{f}'_i = \begin{bmatrix} \Re(\mathbf{f}_i) & \Im(\mathbf{f}_i) \end{bmatrix} \tag{14}$$

of filtered image vectors $\mathbf{f}_i$ and $\mathbf{f}_j$ for each character to be a robust indicator of image discrepancies. We use

$$\kappa_{ij} = 1 - \frac{\mathbf{f}'_i \cdot \mathbf{f}'_j}{\sqrt{\mathbf{f}'_i \cdot \mathbf{f}'_i}\sqrt{\mathbf{f}'_j \cdot \mathbf{f}'_j}} \tag{15}$$

as a distance measure, which has range $[0,2]$. If ρ is the angle between the two vectors $\mathbf{f}'_i$ and $\mathbf{f}'_j$, the distance is related by $\rho = 1 - \arccos \kappa$. When the distance is small the characters are very similar, but when large they are dissimilar. Using the information I^S, we add the factors

$$\log f^S_{ij}\left(y_i, y_j, \mathbf{x}; \boldsymbol{\theta}^S\right) = \delta(y_i, y_j)\, \boldsymbol{\theta}^S \cdot F_{ij}(\mathbf{x}), \tag{16}$$

where $\delta(\cdot,\cdot)$ is the Kronecker delta, and

$$F_{ij}(\mathbf{x}) = \begin{bmatrix} -\ln(\kappa_{ij}) & \ln(2-\kappa_{ij}) & 1 \end{bmatrix} \tag{17}$$

is a vector of basis functions that transform the distance κ_{ij} between two character images in $\mathbf{x}$. The first two functions each have a distance range boundary as an asymptote, and the last is a bias term. Thus, the first weight in the parameter vector $\boldsymbol{\theta}^S$ establishes a high compatibility reward for small distances, the second weight a low compatibility penalty for larger distances, and the bias helps (in conjunction with the first two) establish the crossover point. This is qualitatively similar to the inverse of the sigmoid function with a scaled range, except that it is no longer symmetric about the zero-crossing; see Figure 4. Once again, we note that the function (16) has the general log-linear form as f^A, but we present its more compact version here.

3.4 Lexicon Model

A lexicon is a useful source of high level information that can be applied to recognition. We propose another set of factors for our model that incorporates lexicon information. First, we add auxiliary unknowns that represent lexical decisions. We then add two new factors involving these and the character unknowns. The first factor is simply a bias determining how likely it is *a priori* for a given string to be from the lexicon. The second is a simple binary function that connects all the constituent characters of a word to the lexical indicator. Although this factor is simple in appearance, a naïve implementation would present a great deal of difficulty for common message-based approximate inference methods, such as BP. Fortunately, the form of this particular function makes the implementation much easier, though still linear in the lexicon size. This can be problematic when the lexicon is large, therefore we use a sparse message passing scheme for a lexical model that avoids most of the overhead required with no loss of accuracy on our data. In the remainder of this section, we introduce the new lexical factors, followed by the specialized message passing scheme for inference in the resulting model.

3.4.1 Lexicon Factors

To represent the lexical decision, we introduce auxiliary unknowns $\mathbf{w} = w_A w_B w_C \ldots$ that, for each word in the string $\mathbf{y}$, indicates $w_k \in \{0,1\}$ whether it is present in the lexicon L. For notational clarity, we use numerical indices for the character unknowns $\mathbf{y}$ and alphabetical indices for the lexical/word unknowns $\mathbf{w}$. Let C be the set of unknowns relating to a single word unit; such a set will contain the indices of some letters $\mathbf{y}$ and one entry from $\mathbf{w}$. The factor relating the lexicon, the predicted string, and the lexical decision, is a simple binary function

$$f^W_C(\mathbf{y}_C, w_C) = \begin{cases} 0 & w_C = 1 \wedge \mathbf{y}_C \notin L \\ 1 & \text{otherwise,} \end{cases} \tag{18}$$

where we have written w_C to indicate the value of the sole index of $\mathbf{w}$ present in C. Thus, the corresponding factor (18) is zero only when w_C indicates the string is a word but $\mathbf{y}_C$ is not found in the lexicon. This tautological factor simply represents the proposition

$$w_C = 1 \;\Rightarrow\; \mathbf{y}_C \in L \tag{19}$$

and would not be much use were it not for the fact the value of w_C is unknown. The indicator w_C could help control other aspects of interpretation in the model. For instance, we might want to disable the influence of the local language compatibilities when $w_C = 1$; no matter how unlikely the word *yukky* is[1], it is in the lexicon and should not be discounted for its unusual bigrams during recognition.

The other new factor is a simple term biasing the preference for strings to be drawn from the lexicon

$$\log f^L_k\left(w_k; \theta^L\right) = (1 - w_k)\,\theta^L. \tag{20}$$

1. Under a bigram model trained on a corpus of English text, the word is actually the least likely from a large lexicon.

This function also has a general log-linear form, but we present here its interpretable compact form, so that the single parameter θ^L can be thought of as penalty for non-lexicon predictions.

Introducing these two new classes of factors f^W and f^L will be reflected by conditioning on information I^W and I^L. The factor graph for a model including these factors appears in the bottom of Figure 2. In the next section, we describe more about how the two new compatibility functions (18) and (20) affect inference in the model and introduce the application of a sparse inference technique for making predictions using loopy BP.

3.5 Inference with the Lexicon Factors

Inference, even approximate inference, in the model proposed above might be computationally taxing in general. The sum-product algorithm involves computing local marginals of factors, which is generally much easier than the more global marginalization desired. However, the complexity of marginalizing the lexicon word factors f^W grows *exponentially* with the length of the word. For instance, with a six letter word in a 62 character alphabet, each iteration of message passing would require a sum over 62^6 or nearly 57 *billion* strings.

Fortunately, the on/off "gating" behavior of the function f^W allows us to take advantage of its special form. The effect is that when $w_C = 1$, the "product" in the sum-product equation only needs to be summed over words in the lexicon. For the case when $w_C = 0$, it is summed over all strings, but the sums over constituent characters become independent. This means we can make the calculation a much more efficient product of sums. Thus, the special form (18) makes the inference calculation linear in the size of the lexicon or the character alphabet, rather than exponential in word size. The computational expense of a six letter word drops from billions of possible strings to just a few thousand lexicon words.

As a concrete example, consider the three letter word $y_5 y_6 y_7$ with lexical decision unknown w_B in the right-hand portion of the bottom graph in Figure 2. The general form of a factor-to-node message (cf. Eq. 6) is a product of the factor times the messages to that factor from all its arguments except the message recipient. This product is then summed over all those arguments leaving a function whose value is dependent on the recipient node. To calculate the message from a lexicon factor to the character y_5, we may split the marginalization (the summation of all unknowns except y_5) into two cases, one where the string is a lexicon word and another when it is not: the two values for w_B. For the lexicon factor we are calling C, the specialized form of the message from C to the character y_5 has the form

$$\begin{aligned} m_{C\to 5}(y_5) = \textstyle\sum_{\{y_5 y_6 y_7 \in L | y_5\}} \Big(& f_C^W(y_5, y_6, y_7, w_\mathrm{B} = 1) * \\ & m_{6\to C}(y_6)\, m_{7\to C}(y_7)\, m_{\mathrm{B}\to C}(w_\mathrm{B} = 1) \Big) + \\ & m_{\mathrm{B}\to C}(w_\mathrm{B} = 0)\,. \end{aligned} \tag{21}$$

We first separate the sums for the two values of w_B. Because the factor $f_C^W(\mathbf{y}_C, w_C)$ is zero whenever the argument $\mathbf{y}_C$ is not in the lexicon but $w_C = 1$, the sum can be restricted from all values of $\mathbf{y}_{C\setminus\{5\}}$ to the portion of the lexicon that agrees with the argument value y_5 when $w_\mathrm{B} = 1$. Furthermore, when $w_\mathrm{B} = 0$, the factor is always one. In the last line, we may push the sums over each character value y_6 and y_7 in against the corresponding messages. Because these messages are normalized to sum to one in practice, these terms are dropped, leaving us with a relatively simple sum over a subset of lexicon terms. Calculating the message to character 5 for all values of y_5 involves a sum over all lexicon words of the appropriate length. Messages to other character nodes will have the same form, with the number of node-to-factor messages in the product depending on the length of the word.

Only two values need to be computed for messages from the factor to the word indicator w_B. When the string is not a lexicon word ($w_\mathrm{B} = 0$), the value of the factor is always 1, and the sums over the remaining unknowns in (6) may be pushed inside the product against their corresponding message terms. This results in a product of node-to-factor message sums. Since the messages are normalized, the product (and thus the message value) is simply a constant 1.

When the string is a lexicon word ($w_\mathrm{B} = 1$), the product of messages must only be evaluated at lexicon strings because f_C is zero when the string is not in the lexicon:

$$\begin{aligned} m_{C\to\mathrm{B}}(w_\mathrm{B} = 1) = \textstyle\sum_{y_5 y_6 y_7 \in L} f_C^W(y_5, y_6, y_7, w_\mathrm{B} = 1) * \\ m_{5\to C}(y_5)\, m_{6\to C}(y_6)\, m_{7\to C}(y_7)\,. \end{aligned} \tag{22}$$

4 Sparse Belief Propagation

Although the sums for belief propagation (21) and (22) have a complexity linear in the size of the lexicon, they can still present a computational drag in practice. The top-down information is very important for accurate recognition, so we use a bottom-up scheme to speed the recognition process. Pal et al. [16] propose a probabilistically motivated sparse inference method that simplifies the message passing calculations. The central idea is to reduce the number of summands in such factor-to-node messages by creating zeros in the node-to-factor messages.

At every node, a belief state, or local approximate marginal probability, is represented by the normalized product of messages to that node from its adjacent factors

$$b_i(y_i) \propto \prod_{C \in \mathcal{N}(i)} m_{C\to i}(y_i)\,. \tag{23}$$

During each iteration of loopy BP, each factor combines information from its adjacent node arguments and returns updates to them. As described above, the update for the lexicon factor involves a sum over every word in the lexicon (of the appropriate length), even those words containing characters with low probabilities. We may therefore desire to eliminate these unlikely lexicon words from consideration during the belief update stage. The well-motivated approach given by Pal et al. is to revise the local beliefs such that the largest number of the lowest probability states are given zero probability, subject to a constraint on the divergence of the sparse belief from the original. In other words, consider the fewest number of characters while staying close to the original beliefs. Employing this strategy, we expect to greatly reduce the amount of lexicon scans for a given query.

If b_i represents the marginal belief for node i in the graph, our goal is to compress this distribution to b'_i such that it has the maximum number of zero entries, subject to a divergence constraint:

$$\begin{array}{ll} \text{maximize} & \sum_{y_i \in Y_i} \delta\left(b'_i\left(y_i\right), 0\right) \\ \text{subject to} & \mathrm{KL}\left(b'_i \parallel b_i\right) \leq \varepsilon, \end{array} \tag{24}$$

where

$$\mathrm{KL}\left(b'_i \parallel b_i\right) = \sum_{y_i \in Y_i} b'_i\left(y_i\right) \log \frac{b'_i\left(y_i\right)}{b_i\left(y_i\right)} \tag{25}$$

is the Kullback-Leibler divergence between the original and compressed beliefs. This can easily be accomplished for each node in time $O\left(|Y_i| \log |Y_i|\right)$ by sorting the beliefs and calculating the log cumulant. Once the sparse belief b'_i is calculated, the messages to the factors $m_{i \to C}$ (cf. Eq. (5)) are compressed to respect the sparsity of b'_i and re-normalized.

These sparse node-to-factor messages are then subsequently used to calculate the reverse factor-to-node messages. The practical effect of sparse BP is that certain characters are temporarily eliminated from consideration. For instance, the visual and contextual evidence for y_7 to be a `t` may be so low that it can be assigned a zero belief without greatly changing the current local marginal. When this happens, we may eliminate summands for any word whose second character is `t` in the messages (21) and (22). Taken together, pruning highly unlikely characters reduces the lexicon under consideration from tens of thousands of words to just a few, dramatically accelerating message passing-based inference.

In the original work on sparse BP, only a linear chain graph was used. This topology permits exact inference and requires messages only be passed once in each direction. Here, a loopy variant is used. We note that depending on the order of operations, characters need not be strictly eliminated from possibility when a sparsifying step is taken. Specifically, if outgoing messages to factors are made sparse in agreement with the compressed disribution b'_i, this only means that terms are dropped from the summation used to calculate messages to other nodes. The return message is not sparse in general. Thus, using sparse methods to "eliminate" characters means only that we lose the influence of the dropped character hypotheses upon their logically dependent nodes. The final belief at a node (from which predictions are made) is calculated using the most recent incoming messages from the factors, which are not generally sparse. Therefore we have not necessarily committed to a mistaken elimination of correct character hypotheses. In fact, in some of our experiments, certain character hypotheses are restored as information propagates through the graph.

The information-theoretic criterion for pruning states stands in contrast to that of Coughlan and Shen [17]. In their dynamic quantization algorithm, states are eliminated by thresholding the beliefs $b_i\left(y_i\right) \geq \varepsilon$ and restored by keeping states $\mathbf{y}_C$ that have high factor values $f_C\left(\mathbf{y}_C\right) \geq \varepsilon$. The former criterion may not be stable when the marginal distributions are relatively flat, having many states with equally low probability. The latter criterion may not accurately reflect information from elsewhere in the model. By contrast, the KL-divergence criterion ensures that a minimum total probability mass is maintained for each node's beliefs, and this is constantly updated as information propagates through the graph.

Fig. 5. Examples of sign evaluation data illustrating (left-right) regular fonts similar to those found in documents, unusual regular fonts, and custom irregular fonts.

5 Experiments

In this section we present experimental validation of our model on sign images containing previously unseen fonts and non-lexicon words. The alphabet of characters recognized, Y, consists of 26 lowercase, 26 uppercase, and the 10 digits (62 total).

We find that adding similarity reduces character recognition error by 19%, whereas using it in a separate stage harms accuracy. Adding the lexicon reduces word recognition error by 35%. Using sparse BP eliminates 99.9% of the lexicon, giving a 12X speedup with no loss in accuracy.

First we describe the data used in our experiments for both training and testing, and then the procedures used to train and evaluate the models. The section concludes with the experimental results and a subsequent analysis and discussion.

5.1 Experimental Data

Because we have such a rich model involving many information sources, there are many corresponding data sets for training, including character images, English text corpora, and a lexicon. We describe these after detailing the nature of the primary evaluation data.

Sign Evaluation Data: Our evaluation data comes from pictures of signs captured around a downtown area. There are 95 text regions (areas with the same font) totaling 215 words with 1,209 characters. Many signs have regular fonts (that is, characters appear the same in all instances) that are straightforward, such as basic sans serif, and should be easily recognized. Other signs contain regular fonts that are custom or rarely seen in the course of typical document recognition. Finally, there are a few signs with custom irregular fonts, where repeated characters have a different appearance. These pose the greatest challenge to the premise that similarity information is useful. Examples of each of these three categories are shown in Figure 5. The signs are imaged without extreme perspective distortion—they are often roughly fronto-parallel. Following the assumptions laid out in Section 3, we have annotated our evaluation data with the approximate bounding boxes for the characters.

Synthetic Font Training Data: We generated images of each character in several commercially available fonts using GIMP.[2] Each image is 128×128 pixels with a font height of 100 pixels (an x-height of roughly 50 pixels). No anti-aliasing was used in the image synthesis and the bounding box of the character is centered in the window. [3]

2. GNU Image Manipulation Program `http://www.gimp.org`.
3. Font images and sign evaluation data are publicly available at `http://www.cs.grinnell.edu/~weinman`.

Text Corpora: A corpus of English text was acquired from Project Gutenberg[4]—82 books including novels, nonfiction, and reference for a total of more than 11 million words and 49 million characters (from our 62 character alphabet).

Lexicon: The lexicon we use is derived from SCOWL[5] and contains 187,000 words including proper names, abbreviations, and contractions. Because our current model does not account for word frequency in its lexical bias, we only use those words in the list up to the 70th percentile of frequency for our lexicon.

5.2 Experimental Procedure

In this section we describe the procedure used for training and evaluating our model. We first outline the nature of the overall model parameter training followed by details of training for each component of the model. The section concludes with a brief description of how the model is applied to the actual image data for evaluation.

5.2.1 Model Training

The model parameters $\boldsymbol{\theta} = \left[\ \boldsymbol{\theta}^A \quad \boldsymbol{\theta}^B \quad \boldsymbol{\theta}^C \quad \boldsymbol{\theta}^S \quad \theta^L\ \right]$ are learned from the data outlined above. Typical parameter estimation procedures in such discriminative joint models requires labeled data involving all the information at once. In other words, training data should be like the testing data.

The parameters $\boldsymbol{\theta}^A$, $\boldsymbol{\theta}^B$, $\boldsymbol{\theta}^C$, and $\boldsymbol{\theta}^S$ are each learned independently in the piecewise, decoupled fashion described in Section 2.2, while the lexicon bias parameter θ^L is chosen by cross-validation. Next, we detail the training procedures for each of these parameter sets.

Appearance Model: The character image appearance model parameters $\boldsymbol{\theta}^A$ are trained on 200 fonts, and 800 fonts are used as a validation set. We use a Laplacian prior [27], [28]

$$p(\boldsymbol{\theta} \mid \alpha, I) \propto \exp\{-\alpha \|\boldsymbol{\theta}\|_1\}, \tag{26}$$

where $\|\cdot\|_1$ is the ℓ_1 vector norm. The value of hyperparameter α that yields the highest likelihood on the validation set is the one used for optimizing the posterior for $\boldsymbol{\theta}^A$.

The filter outputs for the 128×128 training images are downsized by a factor of four to 32×32. Although some information from the highest frequency filters is lost, this reduces the dimensionality of $\boldsymbol{\theta}^A$ by a factor of 16.

The evaluation data is far from having perfect contrast (a nearly 0/1 binary image). As a very simple alternative to a more elaborate contrast normalization scheme, we scale the training images so that the contrast (absolute difference between background and character intensity) is 0.3.

Language Model: To avoid the need for performing inference on large chains of text, we use piecewise training (c.f., 2.2) to approximate the likelihood. The approximation is especially advantageous for the bigram and case switch models (11), (12), and (13), which do not depend on an observed image. Thus, training instances may be collapsed into unique cases and weighted by their frequency. For example, the corpus of 49 million characters contains nearly 780,000 occurrences of the bigram `th`. Rather than doing inference on the entire chain of text with an exact method, we need only do inference once in a two-node chain for `th` and count it 780,000 times.

The books were split into two sets of roughly equal size, one for training and one for validation. The (case-insensitive) bigram counts were taken for each set, and the value of the hyperparameter α for the Laplacian prior (cf. Eq. (26)) that yields the highest likelihood on the validation set is the one used for optimizing the posteriors for $\boldsymbol{\theta}^B$ on the entire corpus.

In practice, we found that enabling the language model, regardless of the value of the auxiliary word indicators $\mathbf{w}$ improved accuracy over disabling it whenever the corresponding $w_k = 1$. Our results reflect this aspect of the model.

Case model parameters $\boldsymbol{\theta}^C$ use a uniform prior.

Similarity Model: Because the function f^S is one whenever its two character arguments have different labels and otherwise has a functional value parameterized by $\boldsymbol{\theta}^S$ (displayed in Figure 4), we may equivalently learn the parameters for a function f^S that takes only a single argument y with a label of Same or Different. The piecewise training approximation described above follows naturally because these character pairs are completely decoupled from any related stream of text.

To learn the similarity parameters $\boldsymbol{\theta}^S$ we generated pairs of the same character (in the same font) and pairs of different characters (also in the same font) with the following procedure. First, we select a font and a character uniformly at random. To produce a similar character, we generate a random linear transformation with rotation $\theta \sim \mathcal{N}(0, 1^\circ)$, scale factors $\sigma_x, \sigma_y \sim \mathcal{N}(1, 0.01)$, and skew factors $\rho_x, \rho_y \sim \mathcal{N}(0, 0.005)$. This transformation is then applied to the original image. To produce a dissimilar pair, a different character is chosen uniformly at random. We choose a different character from the same font, having assumed the input is from a single font. Such characters are likely to be more similar than different characters from different fonts, allowing a better and more appropriate threshold to be learned. Additive Gaussian noise $\varepsilon \sim \mathcal{N}(0, 0.01)$ is added to the original and transformed images prior to Gabor filtering. Unlike for the appearance model, the full-size (128×128) filter outputs are used to calculate the distance κ_{ij} between images. The finer details are useful for these comparisons, and the dimensionality is not an issue since f^S only has three parameters.

For optimal predictive discrimination, the ratio of same to different pairs in the training data should be the ratio we expect in testing data. Toward this end, we sample small windows of text from our corpus. The window length is sampled from a geometric distribution with a mean of 10 characters and length at least 3; these parameters are chosen based on our expectation of sign contents. In 10,000 samples, the same/different ratio is consistently about 0.057. This ratio controls the relative number of similar and dissimilar pairs we generated (100,000 total).

Similarity model parameters $\boldsymbol{\theta}^S$ use a uniform prior.

Lexicon Model: We found acceptable performance for models conditioned on I^A, I^B, I^C, I^S with corresponding parameters found in the decoupled piecewise fashion detailed above. One way of doing this for the bias parameter θ^L is to

4. `http://www.gutenberg.org`.
5. `http://wordlist.sourceforge.net`.

use our English corpus, using each word from the text as a training instance with $w = 1$ if it is in our lexicon L and $w = 0$ otherwise. We found that decoupling the learning of θ^L in this way does not yield a strong enough lexical bias to improve results as originally hoped, so we turn to a cross-validation strategy to "re-couple" the parameter learning.

To add I^L to the model, we keep all other parameters fixed at their values learned from decoupling. The 95 regions in the evaluation sign data are randomly split into ten subsets. In a ten-fold cross-validation procedure, we iteratively held out one set for testing. Several values of θ^L are used, and the one with the highest word accuracy on the nine training sets is then applied to the test set for evaluation.

We can also force the model to always predict words from the lexicon by adjusting the bias θ^L to $-\infty$. We will use $I^L_{-\infty}$ to indicate such a closed-vocabulary assumption.

Sparse message passing as proposed by Pal et al. [16] was created for BP in a chain-structured graph where a well-defined forward-backward schedule for message passing achieves exact inference. While the graph based on I^A, I^B, I^C is a chain, adding the lexical information I^W makes this graph not only not chain-structured, but cyclic. Thus, the results of BP will not be exact in general. It is only I^W that is truly problematic from a computational standpoint. The other messages—of which there are only a few—only require complexity of at most $O\left(Y^2\right)$, which is substantially less than the messages from the lexical factor. For this reason, we run BP in a phased schedule, only sending any lexical factor to node messages after the others have converged. Once these messages have converged, we have the best possible local marginals on the available information, excepting I^W and I^L. We then use these beliefs for computing the sparsity of the character states $\mathbf{y}$. This sparsity is calculated once, then the lexical information I^W, I^L is introduced, and the same sparsity structure is maintained. Belief propagation then continues until the termination criterion is reached (convergence or an iteration limit).

This phased processing has two advantages. First, because messages are passed within a limited portion of the model until convergence, the beliefs used to calculate sparsity should be more reliable since the available information has flowed throughout the graph. This stands in contrast with the alternative of doing state pruning with the initial beliefs, which will only be based on factors immediately adjacent to the nodes. Longer distance dependencies certainly exist in these types of models, and these could have an effect on the sparsity and correctness of the approximate beliefs. The second and arguably more important advantage is that it avoids the need to make a complete lexical scan required in the messages from the lexical factor. Since the messages are initialized to uniform, the lexical factor merely ends up contributing positional unigrams to the initial belief. This is not worth the cost of the lexical scan and could be modeled directly if we wished. Returning to our initial point, we prefer to use the best available information before incorporating the lexicon.

We use $\varepsilon = 10^{-7}$ as the divergence bound for sparse BP. This corresponds to keeping nearly all of the probability mass ($e^{-\varepsilon}$) for each character. The runtime was sensitive to this, since it controls the amount of pruning, but we found accuracy was not.

TABLE 1
Recognition results (percentages) of the unified model (top) and clustering followed by recognition and voting (bottom) with varying information. Overall character accuracy, false negative (FNR), false positive (FPR), and hit rates (HR) for pairs (see text) are given.

Information	Char. Accuracy	FNR	FPR	HR
I^A	84.04	11.42	0.51	91.07
I^A, I^S	84.04	11.42	0.51	91.07
I^A, I^B	87.92	9.14	0.53	93.81
I^A, I^C	87.92	8.79	0.87	94.03
I^A, I^B, I^C	91.65	6.85	0.66	98.68
I^A, I^B, I^C, I^S	93.22	5.45	0.14	99.26
I^S	-	22.67	0.25	-
$I^S \to I^A$	83.54	7.03	0.69	88.28
$I^S \to I^A, I^B$	87.92	4.39	0.80	91.73
$I^S \to I^A, I^C$	87.76	5.80	1.02	92.72
$I^S \to I^A, I^B, I^C$	91.40	3.69	0.88	97.26

5.2.2 Model Application

Here we add a few additional details of how the evaluation images are processed for the model. The height of the input characters in the evaluation data is normalized so that the font size is roughly that of the appearance training data. Only filter responses from within the annotated bounding box of each character are used when calculating the factors for appearance f^A and similarity f^S; image areas outside the bounding box are zeroed out. Note that Gabor filters are applied to the actual grayscale image; no binarization is performed.

5.3 Experimental Results

Here we describe the performance of several variants of our model on the evaluation data, as well as alternatives from prior approaches to challenging character recognition problems. First, we demonstrate the impact of using similarity information in a unified model for recognition. Then, we investigate how incorporating a lexicon affects results.

5.3.1 Unified Similarity

Prior work using similarity incorporated this information in a processing stage separate from that using character appearance. Here we will compare our unified model to the approach of Breuel [4], [5], where characters are first clustered using the degree of similarity as a distance metric. Following this approach, to cluster letters, we maximize $p\left(\mathbf{y} \mid \mathbf{x}, \widehat{\boldsymbol{\theta}}, I, I^S\right)$ for $\mathbf{y}$ via simulated annealing, initialized at the prediction derived from I^A (the strategy taken by Breuel [4]). The identity of each cluster is then chosen by using the classification of each character derived from other models (sans I^S) as a vote. Ties are broken by choosing the label whose members have the lowest average entropy for their posterior marginal, a strategy that slightly outperforms random tie breaking.

Table 1 gives the results of the unified model using different combinations of appearance information I^A, language information I^B, I^C, and similarity information I^S. It also shows the

Fig. 6. Examples from the sign evaluation data that are read correctly with I^A, I^B, I^C, I^S.

Fig. 7. Challenging signs from the evaluation data that have unique fonts, are hand-painted, or contain three-dimensional effects, real and virtual.

results when the similarity information is used first to cluster the characters, and the other information (used separately) is then used to vote on character identities. Character accuracy is the percentage of characters correctly identified (including case). To evaluate the ability of our model to recognize different instances of the same character in the same font, for intra-sign and intra-font characters we measure:

- **False negative rate:** Percentage of character pairs that are the same but are given different labels.
- **False positive rate:** Percentage of character pairs that are different but are given the same label.
- **Hit rate:** Percentage of character pairs that are the same, given the same label, and correct (correctly labeled true positives).

For the model I^S, only false negative and positive rates may be reported as cluster purity measures. Figure 6 contains examples of signs correctly read, and Figure 7 shows examples from the evaluation set that are more difficult.

5.3.2 Lexicon-Based Model

In addition to the unified similarity model, we also test the effect of the integrated lexicon and the impact of using sparse BP. Table 2 compares the character and word accuracy for our model with varying amounts of information. For comparison, the output of our best lexicon-free model is passed through the spell-checker Aspell, keeping the top suggestion. Figure 8 shows results on example data of varying difficulty, including where corrections were made and errors introduced.

We show in Figure 9 (top row) the histogram of how many characters remain possible after belief compression with sparse BP for several of the models. The elimination of many characters from consideration excludes certain words in the lexicon with characters in particular positions. The resulting reductions in length-appropriate lexicon words are shown in the bottom row of histograms of Figure 9. Different word lengths have differing numbers of possible words in the lexicon, so we give length-specific lexicon size-normalized values. However, to illustrate the raw impact we also give the median absolute size of the resulting lexicon.

Table 3 compares the accuracy of full loopy BP and the sparse variant used to speed up prediction on the two best

TABLE 2
Word and character accuracy with various forms of the model.

Information	Char. Accuracy	Word Accuracy
I^A	84.04	46.05
I^A, I^B, I^C	91.65	75.35
I^A, I^B, I^C, I^S	93.22	78.60
I^A, I^L, I^W	93.63	72.56
$I^A, I^L_{-\infty}, I^W$	91.56	68.84
I^A, I^B, I^C, I^L, I^W	93.88	85.58
$I^A, I^B, I^C, I^S, I^L, I^W$	**94.62**	**86.05**
$I^A, I^B, I^C, I^L_{-\infty}, I^W$	92.39	81.40
$I^A \rightarrow$ `Aspell`	73.78	53.49
$I^A, I^B, I^C \rightarrow$ `Aspell`	89.50	77.21
$I^A, I^B, I^C, I^S \rightarrow$ `Aspell`	90.98	79.07

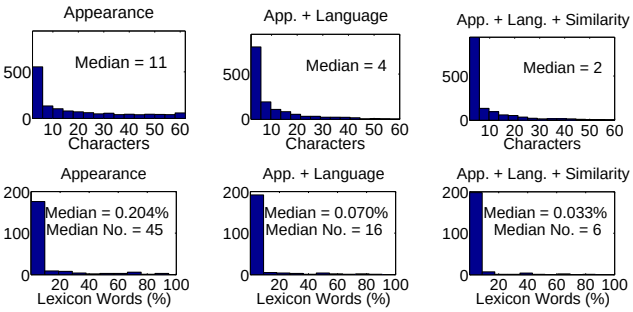


Fig. 9. TOP: Histograms of character state space size after belief compression. BOTTOM: Histograms of lexicon words (percentage) considered after belief compression. LEFT: Appearance only model I^A, I^L, I^W. CENTER: Appearance and language model I^A, I^B, I^C, I^L, I^W. RIGHT: Full model with appearance, language, and similarity $I^A, I^B, I^C, I^S, I^L, I^W$.

models, one with similarity and a lexicon, and one with only the lexicon. We also compare the relative speed of these two models and the different inference techniques in Table 4, as measured by the geometric mean of time per character (to normalize for query length) on the signs [29].

5.4 Discussion

5.4.1 Similarity Model

Figure 6 contains examples of signs correctly read without the lexicon, showing that the features are robust to various fonts and background textures (e.g., wood and brick). Although the number of characters per sign is small compared to OCR applications, adding similarity information undoubtedly improves character recognition accuracy, reducing overall character error by nearly 20% (Table 1). Not surprisingly, most of this improvement comes from greatly reducing the cases when different characters are given the same label (pair false positives).

Perhaps surprisingly, adding similarity information I^S to the simple image information I^A does not alter the results. This is probably because test images have relatively little noise and are mostly difficult due to font novelty and non-fronto-parallel orientations. Therefore, it is expected that the same characters, though novel, would often be given the same label in different locations, due to their logical independence solely with information I^A. However, when other sources of

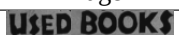
Image	No Lexicon	Lexicon	Forced Lexicon	Aspell
	USED BOOKI	**USED BOOKS**	**USED BOOKS**	USED BIOKO
	Free crecking	Free checking	Free creaking	Frag recurred
	31 BOLTWOOD	**31 BOLTWOOD**	SI BENTWOOD	31 BELLWOOD
	RELAmo3	RELAmo3	**DELANOS**	Reclaim
	HTOR UP5	**HOOK UPS**	**HOOK UPS**	THOR UPI
	MAOUGRY ARAIN	MAOUGRY ANTIN	LATHERY ANTIN	MARGERY AARON
	RERTTRE	RERTTRE	RESTORE	RETRIER

Fig. 8. Example recognition results on difficult data. Correct words indicated in bold. Model examples are I^A, I^B, I^C, I^S (No Lexicon), $I^A, I^B, I^C, I^S, I^L, I^W$ (Lexicon), $I^A, I^B, I^C, I^S, I^L_{-\infty}, I^W$ (Forced Lexicon) and $I^A, I^B, I^C, I^S \rightarrow$ `Aspell` (Aspell).

TABLE 3
Accuracies under sparse and full BP.

	Char. Accuracy		Word Accuracy	
Information	Sparse	Full	Sparse	Full
I^A, I^B, I^C, I^L, I^W	93.88	93.63	85.58	84.19
$I^A, I^B, I^C, I^S, I^L, I^W$	94.62	94.38	86.05	86.05

TABLE 4
Relative speeds of models with full and sparse BP.

Model and Inference	Mean Relative Speedup
No Similarity; Full vs. Sparse	19.53
Similarity; Full vs. Sparse	12.15
No Similarity vs. Similarity; Sparse	0.57

information are introduced to help resolve ambiguity, the similarity information does make a difference because the bigram and case information are based on local context. These can push the beliefs about characters in different directions, even though they tend to look the same, because their contexts are different. Adding the similarity information on top of these other sources ensures that the local context does not introduce a contradictory bias. In the example of Figure 1, adding bigram information pushes the second `e` to an `a` because preference for the `ea` bigram outweighs both `ee` and the character appearance factor. Similarly, adding case information pushes the `l` from being recognized as the upper case `I` to lower case `t`; due to kerning in this italic font, some of the `F` overlaps in the `l`'s bounding box, leaving a little crossbar indicative of a `t`. Finally, adding the similarity information corrects the `l` since it is very different from the final `t`, and corrects the `e`s since they are very similar.

All of the differences in accuracy for the unified model (Table 1) are statistically significant.[6] In particular, adding the similarity information I^S to I^A, I^B, I^C reduces character classification error by 19%. While the reduction of false negatives is not significant with the addition of I^S, the false positives are cut by 79%. When the unified model is compared to the pipelined clustering approach, the differences between I^A, I^B, I^C, I^S and $I^S \rightarrow I^A, I^B, I^C$ are significant for character accuracy, false negative rate, and false positive rate.

The results of clustering the letters prior to recognition appear worse than doing recognition outright with no similarity information. However, unifying all the information available— including similarity— does yield better results than a distinct clustering step. It is interesting that clustering yields fewer false negatives than the unified approach. This is most likely because clusters are not forced to have different labels at the secondary assignment stage. Thus, instances of the same character assigned to different clusters are not forced to have different labels (up to the fact that there are only as many clusters as characters in our alphabet Y). Indeed, if this *were* the case, the false negative rate would be intolerably high. Conversely, the clustering pre-processing step does commit unrecoverable errors by pairing characters that are not the same; subsequent information cannot reduce the false positive rate. This is especially critical because the probability of two characters being the same *a priori* is much smaller than their being different, thus the false positive rate has a greater impact on total errors than the false negative rate.

Some signs in our data set present tremendous difficulty and challenge the assumption that characters of the same "font" appear similar. Some of these are due to rendered warping effects, custom fonts, or inconsistent shadow effects (see Figure 7). Other signs just have unique fonts that are very different from those in the training set.

5.4.2 Lexicon Model and Sparse Belief Propagation

Here we discuss the results of adding the lexicon, some of which are shown in Figure 8. `31` and `BOLTWOOD` are not in the lexicon, so errors arise with the forced lexicon and Aspell models. `DELANOS` is in the lexicon, but the image evidence overpowers the bias in this case; forced to be a lexicon word, it is correctly interpreted. The last two images exemplify some of the more difficult text in our data set.

Incorporating the lexicon factor boosts the character accuracy, but adding the language model (i.e., bigrams) after the lexicon seems to have little impact. However, the word accuracy reveals a 41.5% error reduction with the inclusion of the lexicon. Results do improve over an appearance-only model when words are forced to be from the lexicon, but some proper nouns and numbers in the data are not lexicon words and thus are misinterpreted. Using Aspell fixes some simple errors, but most errors are more complex. Ignoring the character image for poorly recognized words tends to reduce overall character accuracy (since poor suggestions are made). We also experimented with trigrams and word frequencies

6. In all cases, significance is assessed by a paired, two-sided sign test on the accuracy per query; significance is determined by $p < 0.02$.

(i.e., using a word-specific value for U^W), but found no improvement in word accuracy on our evaluation data.

As accuracy gets close to 100%, more data is required for significant improvements to be shown. However, instead of comparing accuracies—the outcome of a decision rule–comparing the likelihoods of the data is more direct way of showing model improvement. When the likelihood of the correct character string is higher in one model than the other, it demonstrates that there is indeed additional information contributed. In this case, when the similarity is added to a model already using a lexicon, the log likelihood ratio improvement is significant (the character accuracy increases from 93.88 to 94.62). The average improvement is

$$\left[\prod_{k=1}^{N} \frac{p\left(\mathbf{y}^{(k)} \mid \mathbf{x}^{(k)}, \widehat{\boldsymbol{\theta}}, I, I^A, I^B, I^C, I^S, I^L, I^W\right)}{p\left(\mathbf{y}^{(k)} \mid \mathbf{x}^{(k)}, \widehat{\boldsymbol{\theta}}, I, I^A, I^B, I^C, I^L, I^W\right)}\right]^{\frac{1}{N}} \approx 3.87. \quad (27)$$

In other words, the data is nearly four times as likely when we add similarity information. This significantly moves the probabilities in the "right direction" relative to the decision rule.

Sparse BP speeds the lexicon integration by eliminating characters from consideration after belief compression (Figure 9). This results in a 99% reduction of candidate lexicon words overall. We must consider different lexicon words for strings of different lengths. The median elimination of candidate words for each string was 99.97% (Figure 9), or just 6 remaining candidates when not normalized for the differing sizes of the original candidate lists. Table 3 shows that using sparse BP yields no significant difference in accuracy. However, there is a very large speed improvement (Table 4), from about 1.36s per character to 0.11s in the complete model.

With sparse BP, adding the similarity information slightly increases the (already greatly reduced) inference time because there are now more factors to pass messages among. Fortunately, the additional similarity information does make character beliefs more certain, allowing more characters and lexicon words to be pruned (Fig. 9). This keeps the additional message passing overhead to a minimum while providing the benefit of a more accurate model.

6 Conclusions

We have laid out a general framework for recognition that is probabilistically well-motivated and can accept long range information in a unified fashion. The conceptual advantage provided by discriminative Markov models easily allows one to imagine and implement a relationship among the unknowns.

Our principal contributions are as follows. First, we have constructed a model that allows unified processing of several important pieces of information (appearance, language, similarity to other characters, and a lexicon). Second, we show how a similarity function can be learned and integrated so that recognition is improved and more consistent with small samples of novel fonts. Finally we have proposed a simple construction that incorporates a lexicon into the model and facilitated its use by applying principled sparse methods.

The basic discriminative framework for character recognition is not new, but it has typically been relegated to individual characters. Language information is usually employed after recognition in a post-processing clean-up. Most prior models integrating language with recognition have been generative, whose independence assumptions often prohibit them from using richer features of the data (observations). A recent exception by Jacobs et al. [10] is a discriminative model, but forces recognition output to be lexicon words. In contrast, our model allows a smooth trade-off between the interpretation of a string as a known word, or some other string.

Classifier adaptation is a useful strategy for recognition. However, when recognizing signs or scene text, there is a scant amount of data, and it is generally insufficient for reliable use with the existing methods for coping with novel typefaces. Our recognition strategy improves on two issues lacking in previous approaches. First, by simultaneously incorporating character identity and similarity information in a unified model, we eliminate the need for distinct clustering/recognition steps and the potential for unrecoverable errors. Second, we treat similarity and dissimilarity as two sides of the same issue, which prevents dissimilar characters from being given the same label.

It has long been known that the use of a lexicon can improve recognition accuracy. Although some computational tricks exist, the size of a lexicon can often be prohibitive for processing that integrates recognition, rather than using it as a post-processing step. Our model provides a natural, practical testbed for the sparse inference methods proposed by Pal et al. [16] for acyclic models. This has the advantage over the traditional approach, which is to prune to one possibility for higher-level processing or to use a more *ad hoc* method to consider a reduced number of alternatives. By eliminating characters from outgoing messages in a principled fashion, we are able to drastically reduce the size of the lexicon that is used for a given query. This does not necessarily mean that characters are eliminated from possibility, since the incoming messages—from which beliefs are calculated—are not generally sparse. We have also introduced lexical decision into a model that also includes other important linguistic cues, such as bigrams.

In this article, we have presented a model for character recognition that ties together several important information sources. We have shown that the unified model clearly improves results over pipelined processing. No doubt many opportunities exist to add other information sources. A richer character recognition model could easily be incorporated to boost accuracy, and higher order *n*-grams for both characters *and* words could be added. All manner of language models could be considered, and there is likely much mileage to be gained by integrating these with the recognition process, rather than using them as post-processors.

Acknowledgments

The authors thank Chris Pal and Charles Sutton for helpful discussions on sparse BP and approximate inference. This work was supported in part by The Central Intelligence

Agency, the National Security Agency, and National Science Foundation under NSF grants IIS-0100851, IIS-0326249 and IIS-0546666.

REFERENCES

[1] Jerod J. Weinman and Erik Learned-Miller, "Improving recognition of novel input with similarity," in *Proc. Conf. on Computer Vision and Pattern Recognition*, June 2006, pp. 308–315.

[2] Jerod J. Weinman, Erik Learned-Miller, and Allen Hanson, "Fast lexicon-based scene text recognition with sparse belief propagation," in *Proc. Intl. Conf. on Document Analysis and Recognition*, Sept 2007, pp. 979–983.

[3] Tao Hong and Jonathan J. Hull, "Improving OCR performance with word image equivalence," in *Symposium on Document Analysis and Information Retrieval*, 1995, pp. 177–190.

[4] Thomas M. Breuel, "Classification by probabilistic clustering," in *Proc. Intl. Conf. on Acoustics, Speech, and Signal Processing*, 2001, vol. 2, pp. 1333–1336.

[5] Thomas M. Breuel, "Character recognition by adaptive statistical similarity," in *Proc. Intl. Conf. on Document Analysis and Recognition*, 2003, vol. 1, pp. 158–162.

[6] John D. Hobby and Tin Kam Ho, "Enhancing degraded document images via bitmap clustering and averaging," in *Proc. Intl. Conf. on Document Analysis and Recognition*, 1997, vol. 1, pp. 394–400.

[7] Sanjiv Kumar and Martial Hebert, "Discriminative random fields," *International Journal of Computer Vision*, vol. 68, no. 2, pp. 179–201, 2006.

[8] Keith Rayner and Alexander Pollatsek, *The Psychology of Reading*, Prentice-Hall, Englewood Cliffs, NJ, 1989.

[9] W. W. Bledsoe and I. Browning, "Pattern recognition and reading by machine," in *Proc. of Eastern Joint Computer Conf.*, 1959, pp. 225–232.

[10] Charles Jacobs, Patrice Y. Simard, Paul Viola, and James Rinker, "Text recognition of low-resolution document images," in *Proc. Intl. Conf. on Document Analysis and Recognition*, 2005, pp. 695–699.

[11] Céline Thillou, Silvio Ferreira, and Bernard Gosselin, "An embedded application for degraded text recognition," *Eurasip Journal on Applied Signal Processing*, vol. 13, pp. 2127–2135, 2005.

[12] R. Beaufort and C. Mancas-Thillou, "A weighted finite-state framework for correcting errors in natural scene OCR," *Proc. Intl. Conf. on Document Analysis and Recognition*, vol. 2, pp. 889–893, 2007.

[13] DongQing Zhang and Shih-Fu Chang, "A Bayesian framework for fusing multiple word knowledge models in videotext recognition," in *Proc. Conf. on Computer Vision and Pattern Recognition*, 2003, vol. 2, pp. 528–533.

[14] Simon. M. Lucas, Gregory Patoulas, and Andy C. Downton, "Fast lexicon-based word recognition in noisy index card images," in *Proc. Intl. Conf. on Document Analysis and Recognition*, 2003, vol. 1, pp. 462–466.

[15] Marc-Peter Schambach, "Fast script word recognition with very large vocabulary," in *Proc. Intl. Conf. on Document Analysis and Recognition*, 2005, pp. 9–13.

[16] Chris Pal, Charles Sutton, and Andrew McCallum, "Sparse forward-backward using minimum divergence beams for fast training of conditional random fields," in *Proc. Intl. Conf. on Acoustics, Speech, and Signal Processing*, 2006, vol. 5, pp. 581–584.

[17] James Coughlan and Huiying Shen, "Dynamic quantization for belief propagation in sparse spaces," *Computer Vision and Image Understanding*, vol. 106, no. 1, pp. 47–58, April 2007.

[18] John Lafferty, Andrew McCallum, and Fernando Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in *Proc. Intl. Conf. on Machine Learning*. 2001, pp. 282–289, Morgan Kaufmann, San Francisco, CA.

[19] F.R. Kschischang, B.J. Frey, and H.-A.Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 498–519, Feb. 2001.

[20] Charles Sutton and Andrew McCallum, "Piecewise training of undirected models," in *Proc. Conf. on Uncertainty in Artificial Intelligence*, 2005, pp. 568–575.

[21] W. Niblack, *An Introduction to Digital Image Processing*, Prentice-Hall, Englewood-Cliffs, NJ, 1986.

[22] John G. Daugman, "Complete discrete 2-D Gabor transforms by neural networks for image analysis and compression," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 36, no. 7, pp. 1169–1179, 1988.

[23] Da Deng, K.P. Chan, and Yinglin Yu, "Handwritten Chinese character recognition using spatial Gabor filters and self-organizing feature maps," in *Proc. Intl. Conf. on Image Processing*, 1994, vol. 3, pp. 940–944.

[24] Xilin Chen, Jie Yang, Jing Zhang, and Alex Waibel, "Automatic detection and recognition of signs from natural scenes," *IEEE Transactions on Image Processing*, vol. 13, no. 1, pp. 87–99, 2004.

[25] B.S. Manjunath and W.Y. Ma, "Texture features for browsing and retrieval of data," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 9, pp. 837–842, 1996.

[26] Karen Kukich, "Technique for automatically correcting words in text," *ACM Computing Surveys*, vol. 24, no. 4, pp. 377–439, 1992.

[27] Joshua Goodman, "Exponential priors for maximum entropy models," Tech. Rep., Microsoft Research, One Microsoft Way, Redmond, WA 98052-6399, 2003.

[28] Peter M. Williams, "Bayesian regularization and pruning using a Laplace prior," *Neural Computation*, vol. 7, pp. 117–143, 1995.

[29] Philip J. Fleming and John J. Wallace, "How not to lie with statistics: the correct way to summarize benchmark results," *Communications of the Association for Computing Machinery*, vol. 29, no. 3, pp. 218–221, 1986.

Jerod J. Weinman received the B.S. degree in Computer Science and Mathematics from Rose-Hulman Institute of Technology in 2001, and the M.Sc. and Ph.D. degrees in Computer Science from the University of Massachusetts Amherst in 2005 and 2008, respectively. He is an Assistant Professor at Grinnell College, Iowa. His research interests include computer vision and machine learning. He is a member of the IEEE, ACM, ACM SIGCAS, and SIGCSE.

Erik Learned-Miller (previously Erik G. Miller) is an Assistant Professor of Computer Science at the University of Massachusetts, Amherst, where he joined the faculty in 2004. He spent two years as a postdoctoral researcher at the University of California, Berkeley, in the Computer Science Division. Learned-Miller received a B.A. in Psychology from Yale University in 1988. In 1989, he co-founded CORITechs, Inc., where he and co-founder Rob Riker developed the second FDA cleared system for image-guided neurosurgery. He worked for Nomos Corporation, Pittsburgh, PA, for two years as the manager of neurosurgical product engineering. He obtained Master of Science (1997) and Ph. D. (2002) degrees from the Massachusetts Institute of Technology, both in Electrical Engineering and Computer Science.

Allen R. Hanson received the BS degree from Clarkson College of Technology in 1964 and the MS and PhD degrees in electrical engineering from Cornell University in 1966 and 1969, respectively. He is Professor Emeritus in the Computer Science Department at the University of Massachusetts Amherst and director of the Computer Vision Lab. His main research interests are in computer vision, particularly vision systems that are capable of functioning flexibly and robustly in complex, changing environments, artificial intelligence, pattern recognition, and learning. He is the author of numerous technical papers in these areas, has been on the organizing committees of most major vision conferences, and has founded two technology-oriented companies. He is a member of the IEEE, ACM, and AAAI.