Toward Integrated Scene Text Reading

Jerod J. Weinman, *Member*, *IEEE*, Zachary Butler, Dugan Knoll, and Jacqueline Feild

Abstract—The growth in digital camera usage combined with a worldly abundance of text has translated to a rich new era for a classic problem of pattern recognition, reading. While traditional document processing often faces challenges such as unusual fonts, noise, and unconstrained lexicons, scene text reading amplifies these challenges and introduces new ones such as motion blur, curved layouts, perspective projection, and occlusion among others. Reading scene text is a complex problem involving many details that must be handled effectively for robust, accurate results. In this work, we describe and evaluate a reading system that combines several pieces, using probabilistic methods for coarsely binarizing a given text region, identifying baselines, and jointly performing word and character segmentation during the recognition process. By using scene context to recognize several words together in a line of text, our system gives state-of-the-art performance on three difficult benchmark data sets.

Index Terms—Scene text recognition, cropped word recognition, character recognition, discriminative semi-Markov model, image binarization, skew detection, baseline estimation, text guidelines, word normalization, word segmentation

1 Introduction

Text is everywhere. Anthropologist Sir John Goody characterizes it as “the most important technological development in the history of humanity” [1]. While humans have been writing for nearly six millennia, machines have made great strides in reading over the last century. In traditional optical character recognition (OCR), a printed document is scanned to an image and translated into some machine readable text format. Although researchers have made significant progress, machines have yet to match human reading performance.

Now the widespread availability of consumer cameras and the worldly abundance of text have created a new era for machine reading. Scene text recognition (STR) involves finding and reading ambient text in the environment captured by a camera. While traditional document processing often faces challenges such as imaging defects, novel or rare fonts, noise, and unconstrained lexicons, STR amplifies these challenges and introduces new ones such as motion blur, curved layouts, perspective projection, and occlusion among others.

Making a robust, accurate scene text reader is a complex problem involving many details that must be handled effectively (see Fig. 1). In this work, we describe and evaluate a reading system that integrates many of these pieces: a simple region-grouping algorithm (see Section 3) and probabilistic models for coarsely binarizing a given text region (see Section 4), identifying baselines (see Section 5), and jointly performing word and character segmentation during the recognition process (see Section 6). By recognizing several words together in a line of text, our system gives state-of-the-art performance on three difficult benchmark data sets (see Section 7).

Our work makes several contributions to scene text reading. Unlike documents, scene text lines may have just a few words. Still, utilizing collinear text words facilitates improved appearance normalization, an important contribution that significantly improves accuracy. Another contribution is the use of the discriminative semi-Markov model, which integrates learning from several information sources such as character appearance, geometry, and language. Finally, we explicitly incorporate word segmentation for STR, a task made challenging by highly irregular and unconstrained character spacing. Because nearly all modules of the system are probabilistic, we pass forward multiple hypotheses to subsequent modules, delaying final decisions until top-down information has been incorporated.

2 Related Work

Many authors have studied the primary task of finding text in images [2], [3], [4], [5], [6]. The 2003 ICDAR Robust Reading competition did much to spur interest in this area [7]. Although other STR work appeared [8], [2], [3], [9], [10] and a follow-up 2005 contest was organized [11], 6 years passed before anyone would benchmark the open-vocabulary word recognition task of this difficult data set [12]. Others have since followed [13], [14], [15], [16], [17]. In 2011, the data set was revised to reduce (though not eliminate) annotation errors, give tighter bounding boxes, and increase certain forms of variability [18]; the word recognition contest received just three entries, the best performing with a 59 percent word error rate, which this work reduces to 42 percent.

Wang and Belongie introduced a new task for their street view text (SVT) data set [13], [19]. Words in each SVT image are to be recognized from a small lexicon of about 50 words.

• *J.J. Weinman, Z. Butler, and D. Knoll are with the Department of Computer Science, Noyce Science Center, Grinnell College, 1116 8th Avenue, Grinnell, IA 50112. E-mail: weinman@grinnell.edu.*
• *J. Feild is with the Department of Computer Science, University of Massachusetts Amherst, 140 Governors Drive, Amherst, MA 01003. E-mail: jfeild@cs.umass.edu.*

Manuscript received 16 July 2012; revised 24 Mar. 2013; accepted 16 June 2013; published online 26 June 2013.
Recommended for acceptance by R. Manmatha.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMI-2012-07-0538.
Digital Object Identifier no. 10.1109/TPAMI.2013.126.

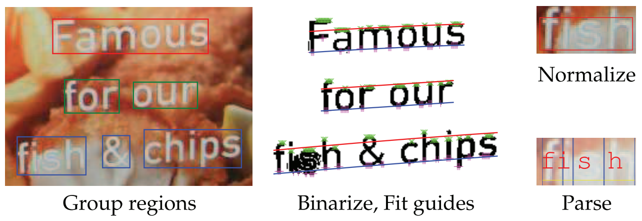

Fig. 1. Detected text regions are collected into lines, coarsely binarized and fit with quadratic text guides. The lines are normalized and passed on for recognition.

Though the images are more intrinsically challenging due to resolution, mosaicing errors, and perspective, the word spotting task is much simpler than the open-vocabulary ICDAR benchmark.

Our prior work assumed character and word boundaries for recognition [20]. To find word boundaries, Neumann and Matas [14], use heuristics of gaps on binarized character regions, while Shivakumara et al. [21] use weak image gradients. Before distances between characters can be measured, characters must be binarized correctly, a significant challenge when noise and low resolution cause both broken and touching characters. Even if characters could be binarized and isolated, word boundaries are not easily predicted because scene text is often less constrained by character kerning and tracking conventions. For example, the intra-word gaps in FIRE are larger than the interword space in MARTIAL ARTS.

To resolve these ambiguities, we integrate word and character segmentation with character recognition [22], [23], giving bottom-up and top-down information flows influence so that low-level segmentation commitments are not made too early and high-level recognition processes need not examine unsupported hypotheses. Several others have since applied similar weighted finite state transducer (FST) variants to the character segmentation and recognition problem. Saidane et al. [12] use a convolutional neural network to predict character boundaries, building a recognition graph with segmentation and letter scores from a convolutional neural network as edge weights. With no language model included, only one letter hypothesis per edge is needed to find the maximal score. Elagouni et al. [16] follow by incorporating a trigram language model. Yamazoe et al. [15] similarly use a weighted FST, with lexicon-constrained paths. Such lexicon constraints are suited to the word spotting task, where Wang and Belongie [13] use a pictorial structures model, a type of weighted FST with quadratic edge scores. Mishra et al. [17] take an intermediate approach by using positional bigram statistics, an approach whose power devolves as the lexicon grows and is only applicable when word segmentations are assumed.

Many of these informal approaches have their roots in the Markov models of speech and handwriting recognition, where segmentation must be integrated. In this work, we use the discriminative semi-Markov model [24], a globally normalized, nongenerative cousin of the variable duration hidden Markov model proposed by Ferguson for speech recognition [25]. Because our probabilistic model has a global normalization factor, all the features, from appearance to language to geometry, and their relative importance can be learned in an integrated fashion, as in the nonprobabilistic setting of Lecun et al. [26].

Other machine perception problems, especially handwriting recognition and speech understanding, are closely related to STR and share many techniques. Integrating lexical recognition or word segmentation requires tracking many hypotheses, which excessively consumes memory. Speech recognition systems commonly reduce this large search space heuristically with Viterbi beam search [27]. In handwriting recognition, Liu et al. [28] combine the three most common beam pruning methods: absolute threshold, relative cost difference, and maximum number of active hypotheses.

Many approaches require characters to be isolated in the image before recognition. Some detect character candidates with sliding windows [13], [17], while others classify connected components [14]. We take a more holistic view by fitting guidelines to coarsely binarized groups of colinear text regions, which may then be normalized for an integrated segmentation and recognition process. We first describe these three important preprocessing steps (see Sections 3-5) before detailing our probabilistic recognition model (see Section 6).

3 Region Grouping

Both texture and connected component-based text detectors can break up a single line of text into multiple detections. While separated regions often occur across word boundaries, weak bottom-up signals due to shadows or blurring within words can also cause separation and undetected characters. Grouping these independent detections together can help later stages in at least two ways. First, fitting guide curves to the text becomes more robust with increased data. Second, the problem of undetected characters can be resolved by postponing the character/no character/space decision to a later stage, when more context is available.

Several authors propose techniques for grouping connected components into text lines for recognition. Models that incorporate local and pairwise component features in an energy-minimization framework are common for handwriting [29] as well as scene text [30], [31]. Others use simpler heuristics and incremental pairwise feature comparisons [32], [14].

Our simple but effective approach is motivated by our benchmarks' input data format: axis-aligned word bounding boxes. While most any of the sophisticated methods above might be used, we simply iteratively link the closest pair of available boxes, subject to a few simple constraints:

1. The horizontal overlap may not exceed 15 percent of the smaller box height.
2. The smaller box height may be no less than 45 percent of the larger box height.
3. The average distance between corresponding points may not exceed 120 percent of the average box height.

The first constraint prevents unrelated but coincident boxes from being linked. The second constraint allows a box with both ascenders and descenders to be linked with a neighbor that may have neither, but otherwise prevents linking

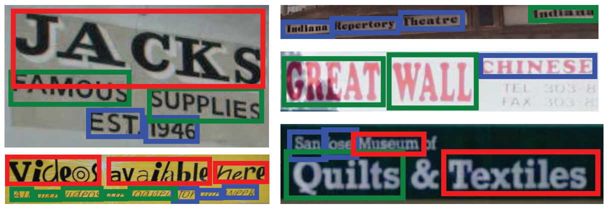

Fig. 2. Word box line grouping examples.

collinear boxes of different font sizes. While some minor false positives do accrue, this constraint helps to filter superscript marks such as ® and ™. The final constraint allows some flexibility for grouping curved or rotated text and word boxes separated by a reasonable amount of space, but it otherwise prevents linking boxes too distant to belong to the same line. The parameters and rules were identified and tuned with the ICDAR scenes training data.

Subject to these constraints, we iteratively link word boxes in ascending order of the average distance between their adjacent, corresponding points (in a left-to-right orthography). Thus, each box is followed by its closest unlinked successor.

As the examples in Fig. 2 show, overlapping regions, rotated text, and long lines can be handled reasonably well. Despite the algorithmic simplicity, quantitative evaluation demonstrates relatively few false positives (0.0145 percent), which are more problematic than the false negatives (6.74 percent); see Table 1. To hedge against false positives, we apply the subsequent steps (image normalization and recognition) both with and without grouping, keeping the best scoring final interpretation for each word.

TABLE 1
Grouping in ICDAR 2011 Test Scenes (1,189 Words)

Collinear Word Rectangle Pairs

		Predicted	
		Positive	Negative
Correct	Positive	429	31
	Negative	11	7535

4 Segmentation and Binarization

The 3D orientation of scene text implies some form of normalization is critical for recognition. Normalization typically requires inferring character geometry from binarized text. Unfortunately, environmental factors such as lighting, low resolution, and noise make accurate bottom-up text binarization difficult. However, a crude binarization often suffices for a geometric analysis, even if it does not segment letters perfectly or has other artifacts ill-suited for recognition.

Given a candidate image region from a text detector, we perform a variety of segmentations and choose the most text-like binarization. After discussing related work, we first describe how to generate candidate region segmentations and then how we select a binarization from among them.

4.1 Related Work

Many approaches to scene text character binarization begin with a clustering segmentation, followed by a classification step to identify text segments. While Kita and Wakahara [33] score all binarizations of a k-means model (i.e., $2^5 = 32$ for $k = 5$) to find the best for a single character, Zeng et al. [34] use a simple heuristic for choosing which of the $k = 3$ modes for an entire word corresponds to text. Cho et al. [35] create a watershed segmentation of a word image, where all segments are jointly classified as part of a character by a CRF. Neumann and Matas [6] independently classify extremal region binarizations as characters with an SVM. Rather than using top-down features of characters to classify segments, Mishra et al. [36] use a bottom-up seeding approach to guess at text/nontext regions before estimating an MRF with latent Gaussian mixture models.

Rather than classify individual connected components or k-means mode combinations as characters, we take a global bottom-up approach to image segmentation followed by a more top-down text binarization.

4.2 Segmentation with Regression Mixtures

A common model for unsupervised color-based image segmentation is the simple Gaussian mixture,

$$p(\mathbf{c} \mid \boldsymbol{\mu}, \sigma, \boldsymbol{\pi}) = \sum_{k=1}^{K} \pi_k p(\mathbf{c} \mid \boldsymbol{\mu}, \sigma, z = k), \tag{1}$$

where $\mathbf{c} \in \mathbb{R}^3$ is a color column vector at some pixel location, which is modeled by a set of K Gaussian densities $p(\mathbf{c} \mid \boldsymbol{\mu}, \sigma, z = k) = \mathcal{G}(\mathbf{c} - \boldsymbol{\mu}_k; \sigma^2)$ with mixture parameters $\pi_k = p(z = k)$ and scalar variances σ^2. The posterior probability $\gamma_k = p(z = k \mid \mathbf{c}, \boldsymbol{\mu}, \sigma, \boldsymbol{\pi})$ induces a segmentation from the most likely component for each pixel, $\hat{z} = \arg\max_k \gamma_k$.

This simple model can fail in scene images where lighting changes across the image region or designers employ color gradients; an accurate segmentation may require many components, large variances in the Gaussian components, or both. A more compact and reliable approach is to model change directly. We follow the approach of Tu and Zhu [37], who use a cubic Bezier surface to model smooth color changes over a region. Rather than a mixture of simple Gaussian distributions (1), we instead have a mixture of regressors [38], with the mean of each component dependent on the pixel location:

$$p(\mathbf{c} \mid \mathbf{x}, \mathbf{B}, \sigma, z = k) = \mathcal{G}(\mathbf{c} - \mathbf{B}_k^\top \boldsymbol{\varphi}(\mathbf{x}); \sigma^2). \tag{2}$$

where $\mathbf{B}_k$ is a $3 \times d$ parameter matrix representing a regression over a higher dimensional representation $\varphi : \mathbb{R}^2 \rightarrow \mathbb{R}^d$ of the pixel coordinates $\mathbf{x} = (x, y)^\top$. While Tu and Zhu [37] use a cubic expansion, making $\mathbf{B}_k$ a 3×16 parameter matrix, we find a quadratic expansion kernel $\boldsymbol{\varphi}(\mathbf{x}) = (x^2, y^2, xy, x, y, 1)^\top$ superior to a linear kernel $\boldsymbol{\varphi}(\mathbf{x}) = (x, y, 1)^\top$, and both an improvement over a simple mixture model. Because our Gaussian model is diagonal, we use the YCbCr color space, whose components are designed to be more independent than those of RGB.

An expectation-maximization (EM) framework finds a local maximum likelihood solution of $\mathbf{B}$ and π with fixed $\sigma = 3/255$. Given $\mathbf{B}$ and $\boldsymbol{\pi}$, the traditional E-step computes updated values of the posteriors $\boldsymbol{\gamma}$. Given those posteriors in the M-step, the mixing parameters $\boldsymbol{\pi}$ are updated in the usual way, while each $\mathbf{B}_k$ may be found as a weighted least-squares

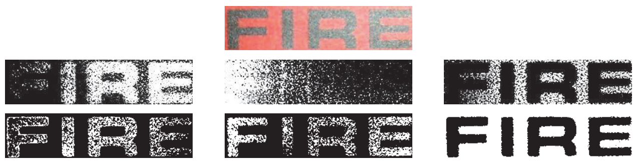
Fig. 3. Component posterior probabilities with $K = 3$ using a simple Gaussian mixture model (top) and a mixture of linear regression models (bottom).

solution to $\mathbf{C} = \mathbf{B}_k^\top \Phi(\mathbf{X})$, where $\mathbf{C}$ is a stacked version of the colors $\mathbf{c}$ for all pixels, and $\Phi(\mathbf{X})$ is the stacked version of corresponding "lifted" pixel locations. The posterior probability γ_k becomes the weight for that pixel location's contribution to the least-squares problem.

To insure against bad local maxima of EM, we learn the constant offset parameters with a standard mixture model (1) before fitting the full regression model parameters. As Fig. 3 demonstrates, a simple Gaussian mixture model is not well suited to lighting changes, while coupling the color values through a latent linear model dependent on pixel location is often sufficient to separate the image components.

4.3 Binarization

Inspired by Kita and Wakahara [33], we choose among several candidate segmentations given by the regression mixtures to yield a final binarization. We fit one mixture of $K = 3$ components and another with $K = 4$. In many cases, the components in the $K = 3$ model correspond to text, background, and mixed pixels. The primary question is how to identify which component(s) correspond to the text. We pose this as a probabilistic classification problem, where a logistic regression scores several features of all binary images induced by the mixture model. The candidate binarization with the highest probability of being text is forwarded to the subsequent stage (guide line fitting). Next, we describe how candidate binary images are generated, the features used to make the classification, and how the classifier is trained.

4.3.1 Connected Component and Binary Image Features

For each mixture component, we create a binary image where a pixel is "on" if the given component is the most probable under the model (2). We also consider the union of pairs of binary images, which allows us to handle strong shadows, dual-color characters and so on. Because the background is sometimes segmented more uniformly than the text, we include the binary complement of each candidate image for consideration. In total, there are six unique images for $K = 3$, plus 20 more candidate binarizations for $K = 4$.

We use two classes of features to represent each binary image: statistics of connected component features and global statistics of the binary image. We measure 14 features for each connected component: normalized area [34], hole/area ratio, compactness, solidity (three features used by Neumann and Matas [6]), eccentricity, normalized major and minor axis lengths, aspect ratio, and Hu's seven invariant moments [39]. Because the number of components varies, we represent the distribution of each component

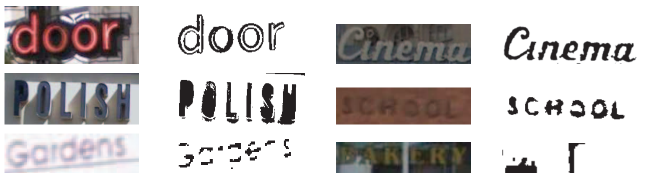
Fig. 4. Binarizations of street view text words. Inferred text is black on a white background.

feature with a fixed set of 11 feature statistics: mean, variance, skew, kurtosis, quantiles at 2.5, 25, 50, 75, and 97.5 percent, and sums of the absolute deviation from the mean and median.

We compute five global features of the binary image: normalized total area, fraction of "on" border pixels, Euler number, and normalized horizontal and vertical range. Finally, we compute the same 11 statistics described above on five functions of the pixels: stroke width (distance from every skeleton point to the nearest "off" pixel), normalized row and column coordinates, and normalized marginals (e.g., column and row sums divided by height and width). All features together form a 225D representation of a binary image.

4.3.2 Text Image Classifier

For training data, we labeled several word image segmentations from the $K = 3$ regression mixture model (2). Additional positive instances came from automatically binarized images of words from the ICDAR 2003 scenes training data provided by Mishra et al. [36].

We use a logistic regression classifier to score each of the 26 candidate images given by the $K = 3$ and $K = 4$ regression mixture models, identifying the image with the highest probability of being text. If none of these images yields a positive classification, we run another mixture with $K = 5$. Considering all components and pairs of components along with the complements yields 30 candidate binarizations for the $K = 5$ model. The binarization with the highest classification score (now among all 56 candidates) is finally chosen as the binarized text image. Trying the more restricted models first prevents a spurious but high-scoring $K = 5$ oversegmentation from being chosen when a simpler model suffices. On the SVT test data, the percentage of binarizations drawn from the $K = 3, 4, 5$ models are 74, 20, and 6 percent, respectively. Of those using the $K = 5$ model, 70 percent are still classified as nontext.

Instead of standard logistic regression, which induces a linear decision boundary in the feature space, we "raise" the features into a quadratic decision space by including all product pairs of the feature vector elements for a total of 25,425 features. To prevent overfitting in this high-dimensional space, we use cross-validation to choose the weight of a sparsity-inducing Laplace prior (ℓ_1 regularization) on the weights [40], which pruned 30 percent of the features.

Fig. 4 above illustrates examples from the SVT data. Common failure modes (cf. bottom row) include thin, undersegmented characters and text-like blobs, in addition to polarity inversion. However, unlike the method of Mishra et al. [36], ours is not sensitive to an initial, bottom-up seeding process to guess which regions are text.

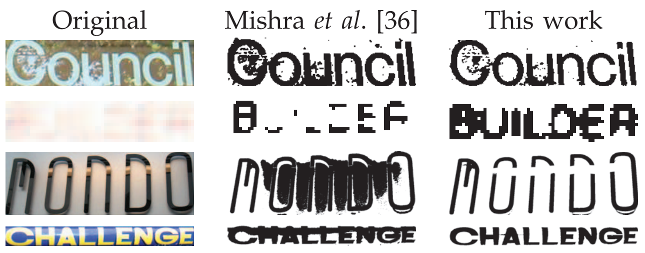

Fig. 5. ICDAR word binarization comparison.

Instead, it waits until more geometric features are available to classify the final segments as text, the advantages of which can be seen in Fig. 5.

Table 2 qualitatively evaluates our technique on the KAIST scene text database English subset with rectangular bounding boxes [41]. Though each method uses the same binarization classifier, segmentations are provided by a simple Gaussian mixture (1) or regression mixtures (2). A paired, two-sided Wilcoxon signed rank test indicates the recall and F1 of the regression mixtures are both significantly better than the simple mixture model; precision differences are not significant.

Finally, we reemphasize that the purpose of this stage is *not* to achieve a binarization worthy of accurate recognition. Rather, we simply want to enable a geometric analysis for word normalization prior to a recognition based on the full color image. Thus any sufficiently advanced binarization algorithm should suffice. Next we describe how these coarse binarizations can improve recognition by facilitating word image normalization.

5 Text Line Normalization

Text in the wild is captured from arbitrary views and exhibits rotation, perspective projection from nonplanar surfaces, and even intrinsically nonlinear baselines (the conventional term for the curve on which the letters rest, whether it is truly linear or not). While it is possible to train a character classifier with examples of these variations, the result is often inherently less discriminative unless invariant features are used or the variations are explicitly modeled rather than being treated as noise. We hope to sharpen performance of our character recognition system by using the binarized image to normalize the text to a rectilinear pose, minimizing the effect of viewpoint variations, as outlined in Fig. 6.

5.1 Related Work

Whether in documents, handwriting, or scenes, normalizing text typically improves recognition accuracy. In document processing, this task usually consists of "skew detection," which simply means inferring the global page rotation (see Hull's survey [42]). In handwriting recognition, Bengio and LeCun [43] fit four tied quadratic curves to individual words. They use local extrema of online pen trajectories as control points for a robust least-squares regression built on a formal probabilistic model. For offline handwriting, Caesar et al. [44] fit four tied straight lines to groups of words, using simple vertical extrema of the binary image as control points. In scene text, Shivakumara et al. [21] perform a simple rotation of a binarized detection. Neumann and Matas [14] assume a planar text surface with two linear guides to estimate foreshortening parameters for normalization. In later work, they follow Caesar et al., fitting four tied lines to binary character vertical extrema with a least median squares regression [45].

TABLE 2
Average Binarization Results for KAIST/English Data [41] (395 Images)

Method	Prec.	Recall	F1
Simple Gaussian Mixture	70.50	86.29	74.30
Linear Regression Mixture	68.53	89.71	75.25
Quadratic Regression Mixture	69.95	90.59	76.76

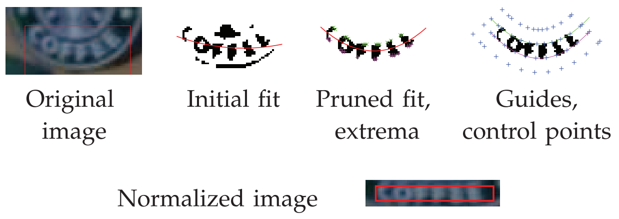

Fig. 6. Word normalization process.

Unlike handwritten text, typefaces exhibit substantial regularity in the locations of the capline (tops of the upper case or capital letters), meanline (tops of the lower case letters), and descender line for a given font size and baseline. Thus for many recognition tasks, identifying only a pair of the guides is sufficient for normalization; a character appearance model can easily handle the small variation remaining. Our algorithm infers two text guides from the binarized image, one for the baseline and one for character tops, which may be either the meanline or the capline.

Several important requirements must be satisfied in order for any algorithm to perform robustly. First, because binarization is imperfect, the guide curve inference must be robust to noise, outliers, false positive and missing characters. Second, because text can appear in practically any orientation, the algorithm must be flexible enough to handle rotated, curved, and nonparallel guides while preventing overparameterized fits to noise or small amounts of text data. To satisfy the first requirement, we use a probabilistic mixture explicitly modeling outliers. For the second requirement, we model the guides as two loosely coupled curves. A second-degree polynomial is sufficient to handle the most common variations found in scene text, while purely linear baselines are surprisingly rare due to lens distortion. The loose coupling means that while the guides tend to vary together (i.e., for rotated text), our model can still handle perspective convergence (e.g.,) and independent upper and lower guides (e.g.,).

5.2 Probabilistic Regression Formulation

In our formulation, we have only the binary image and must infer which points correspond to the extrema that the guides intersect. Unfortunately, these extrema are defined in a coordinate system relative to the guide curves themselves. Consider the character "o." Under any rotation, the image coordinate system gives an arbitrary point on the boundary as a lower extrema. Forcing a guide through this arbitrary point makes the character rest below the baseline. The converse happens for upper extrema. Prior work is prone to these biases for rotated text. Because we need the guides to infer the extrema and the extrema to infer the guides, we take an approximate, iterative refinement approach to discovering both.

Following Caesar et al. [44], we first find the least-squares quadratic fit to all the points (pixels) in the binarized image. This fit tends to give a good approximation of the text guide shape because most of the pixels are text. However, a simple least-squares regression is not robust, and nontext outliers can significantly bias the results. We therefore discard any connected components this initial curve does not touch and perform another regression on what remains. This secondary fit is usually much closer to the correct guide shape (assuming top and bottom are similar to one another), tends to traverse the characters, and gives us a reasonable, approximately correct coordinate system for finding extrema to which lower and upper guides may be fit.

To find candidate extrema, we consider a series of local coordinate systems aligned with the curve's normal and tangent. At each column, we search along the normal of the secondary fit to find the farthest point from the curve on each connected component intersecting that normal. We then retain only those that are extremal points of the binary image in this local coordinate system. To assess which points are extrema, we consider a neighborhood of 5 pixels to either side (in the tangent orientation) for comparison. While this neighborhood is not scale invariant, results are stable over a wide range of scales.

Following Bengio and LeCun [43], we define a probabilistic model for the coefficients $\boldsymbol{\beta} \in \mathbb{R}^{d+1}$ of each degree d polynomial that describes a text guide. Let $\mathbf{X} = \{\mathbf{x}_i\}$ be the set of observed lower (upper, respectively) points $\mathbf{x} = (x, y)$. Because a given point may be an outlier due to a bad binarization or a descender (ascender, respectively), we model the likelihood of each point as a mixture between a Gaussian and a uniform density:

$$p(\mathbf{x}_i \mid \boldsymbol{\beta}, \epsilon, \upsilon, \rho) = \rho \mathcal{G}(y_i - \boldsymbol{\beta}^\top \boldsymbol{\varphi}(x_i); \epsilon^2) + (1-\rho)\upsilon, \tag{3}$$

where $\boldsymbol{\varphi}(x) = (x^2, x, 1)^\top$ for a $d = 2$ quadratic guide, ϵ describes the error (standard deviation) expected for the fit, $0 < \rho \leq 1$ is the mixing prior for inliers, and υ is the (finite) inverse area of the uniform background model. To promote similarity to some preliminary curve, we incorporate a prior on the guide coefficients:

$$p(\boldsymbol{\beta} \mid \boldsymbol{\alpha}, \boldsymbol{\sigma}) = \mathcal{G}(\boldsymbol{\beta} - \boldsymbol{\alpha}; \boldsymbol{\sigma}^\top \mathbf{I} \boldsymbol{\sigma}), \tag{4}$$

where $\boldsymbol{\alpha} \in \mathbb{R}^{d+1}$ describes the preferred coefficients, with a tolerance vector $\boldsymbol{\sigma} \in \mathbb{R}^{d+1}$. We find the most likely coefficients for the fit according to the posterior probability

$$p(\boldsymbol{\beta}, \rho \mid \mathbf{X}, \boldsymbol{\alpha}, \boldsymbol{\sigma}, \epsilon, \upsilon) \propto \prod_{i=1}^{N} p(\mathbf{x}_i \mid \boldsymbol{\beta}, \epsilon, \upsilon, \rho) p(\boldsymbol{\beta} \mid \boldsymbol{\alpha}, \boldsymbol{\sigma}), \tag{5}$$

fixing the uniform background likelihood at $\upsilon = 10^{-21}$. While a value on the order of the image size would seem more appropriate, in the absence of prior information on ρ, an extreme number is necessary to prevent most all points from being assigned to the background model. The small υ therefore functions as a proxy for a strong bias toward inliers. Although not scale invariant, we fix the error term to just $\epsilon = 0.5$ pixels to promote tight fits rather than guides that run between baseline and descender points. We discuss the deviation tolerances $\boldsymbol{\sigma}$ in the next section, where we describe a multistep process for finding a good local maximum of this posterior (5) for the upper and lower guides.

5.3 Expectation-Maximization Fitting Algorithm

Because the log posterior (5) is not convex, we use an iterative EM approximation to estimate coefficients. As with the mixture of regressors in Section 4.2, calculating guide coefficients $\boldsymbol{\beta}$ in the M-Step amounts to solving a weighted least-squares problem, due to the Gaussian likelihoods and prior.

Bengio and LeCun's model explicitly couples their four fitted guides. However, to avoid bad local maxima, we implicitly couple the inference of our two guides through a sequential process that first infers the baseline curve, using the result to bias the upper curve. To fit the baseline, we must set the hyperparameters $\boldsymbol{\alpha}$ and $\boldsymbol{\sigma}$. The pruned, secondary regression gives us the higher order coefficients (e.g., α_2 and α_1). With these two terms fixed, α_0 is the least-squares estimate on all the lower extrema points, even though some may be outliers. The corresponding deviation tolerance is $\boldsymbol{\sigma} = (10^{-5}, 10^{-3}, \bar{h})$ where $\bar{h}$ is the average text region rectangle height.

For stability, we then condition the upper curve's coefficients on the baseline estimate, but complications remain: characters may reach either the meanline or capline; a noisy binarization has outliers. Because inliers are typically offset from the baseline by the same amount, we increase our chances of finding a good-fitting upper curve by first clustering the distances from the upper points to the baseline curve with a $K = 4$ Gaussian mixture model. We initialize the means to uniformly spaced percentiles of the distances, discarding the largest and smallest values for stability to outliers. The posterior (5) for the upper curve's coefficients is maximized with EM. We initialize the M-step with the cluster probabilities of the upper extrema points under each mixture component in turn. From these varying initial conditions, we keep whichever result has the highest posterior probability.

The prior coefficient means α_2 and α_1 use the values from the bottom curve, with α_0 an analogous least-squares estimate on the upper extrema. In many circumstances, we expect the baseline curve to be more similar to the upper curve so we can use much smaller tolerances, $\boldsymbol{\sigma} = (10^{-10}, 10^{-7}, 1)$ for the ICDAR scenes and a slightly more forgiving $\boldsymbol{\sigma} = (10^{-9}, 10^{-6}, 1)$ for the SVT data, which exhibits more variation.

To control model complexity, we use MacKay's "Occam factor" [46] to approximate the Bayesian evidence for linear $d = 1$ and quadratic $d = 2$ models. We select the model degree with the highest approximate posterior probability when the coefficients $\boldsymbol{\beta}$ are marginalized.

Fig. 7 shows common failure modes: too few data points with descenders (or ascenders), ascender/descender drift,

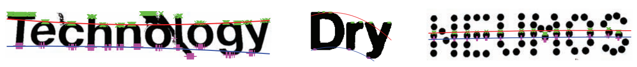

Fig. 7. Guide fitting failure modes.

and broken characters, in addition to poor binarizations with too many distractors.

Once we infer the pair of guide curves, we normalize the word images to horizontal, straight line guides using a thin-plate spline and bicubic interpolation. For each of several evenly spaced points along the baseline curve, we find the baseline normal's intersection with the top curve. To cover ascenders and descenders, we extend two more points along the normal above and below the curves. These four points form a vertical line in the normalized image, with corresponding points aligning horizontally. Note that the control points are established using only the curve geometry, with no underlying image information. Fig. 8 gives further examples of fits and their normalized results.

Sometimes the upper curve settles on the meanline, e.g., , and sometimes on the capline, e.g., . Rather than attempt further bottom-up inferences at this stage, we assume either interpretation could be correct and recognize the words under both interpretations, keeping whichever scores higher. We could also enhance this more Bayesian approach by sampling multiple modes of the guide probabilities to create several normalized image candidates for recognition.

6 Text Line Recognition

Section 4 demonstrated the difficulties of binarizing low-resolution characters, which often results in a single connected mass or characters broken into multiple segments. Without more information about the characters, a single algorithm or parameter setting is unlikely to correctly binarize all inputs. This section explains our approach to integrating character segmentation with recognition. With the normalization described in Section 5, a 2D curved or rotated text layout is transformed into a 1D representation of the text string, allowing us to use standard linear sequence models for character segmentation and recognition. We extend the approach to find word boundaries, placing the entire process into a probabilistic framework.

If word boundaries are known, we can force the model to interpret a given word image as a lexicon string and return the highest scoring word [47], [13], [17]. However, this method does not apply when word boundaries are unknown. Therefore, in a manner analogous to examining the hypothesis of starting a character at every location, we consider starting a word at every location. The concomitant complexity requires us to borrow sparse beam search tools from speech recognition to eliminate unlikely word hypotheses.

Our model requires no binarization or prior character segmentation. By integrating recognition with segmentation at both word and character levels, we reduce the strict reliance on uninformed bottom-up techniques, avoiding unrecoverable errors. Furthermore, because the model seeks to explain an entire normalized text region, it is not overly sensitive to missed detections of isolated characters in earlier pipeline stages [13], [14], [17].

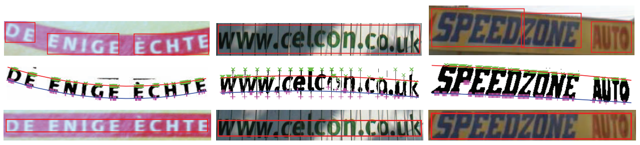

Fig. 8. Guide fitting and normalization examples.

Like many contextual recognition models, ours employs compatibility functions using various information sources to relate label hypotheses to one another and the image features. Although the conventional conditional random field (CRF) is a powerful tool for sequence modeling, the discriminative semi-Markov model [24] is more natural because it not only captures the dependencies between states but also their duration, automatically yielding segmentations. Unlike the generative duration models used in speech [25] and handwriting recognition [48], the discriminatively trained model need not make independence assumptions of the input, allowing us to use global image features.

In the following, we first establish the basic model formalism, then describe how several information sources score a segmentation and labeling, the dynamic programming algorithm for finding the optimal segmentation and labeling, and finally how the model is trained.

6.1 Semi-Markov Model

A segmentation $\mathbf{s}$ induces a sequence of labels $\mathbf{y}$ and a corresponding set of discriminant functions $\{U_C\}_{C\in\mathcal{C}(\mathbf{s})}$. Each segment $s_i = (r_i, t_i)$ with $r_i < t_i$ indicates the start and ending location of a character $y_i \in \mathcal{Y}$, which takes on a label from some alphabet, i.e., $\mathcal{Y} \equiv [\mathtt{A} - \mathtt{Za} - \mathtt{z0} - 9\sqcup]$ where $\sqcup$ is an interword space. Modeling spaces explicitly as a character allows seamless integration of word segmentation with character recognition and segmentation. The conditional probability over segmentations and the labels is an exponential model:

$$p(\mathbf{s}, \mathbf{y} \mid \mathbf{x}, \boldsymbol{\theta}) \propto \exp\left(- \sum_{C\in\mathcal{C}(\mathbf{s})} U_C(\mathbf{s}_C, \mathbf{y}_C, \mathbf{x}; \boldsymbol{\theta}) \right), \tag{6}$$

where $\mathbf{x}$ represents the observed image, and $\mathcal{C}$ is a set of learned discriminant functions that depends on the segmentation. For instance, each segment's unknown y_i will require a function corresponding to a character recognition discriminant. The notation $\mathbf{y}_C$ indicates the subset of the labels that are arguments to U_C.

6.2 Model Features

In this section, we describe how a segmentation $\mathbf{s}$ and labeling $\mathbf{y}$ are scored, given the image $\mathbf{x}$ and parameters $\boldsymbol{\theta}$. Compatibility functions representing appearance and language information aid recognition. Additional functions capturing layout information aid in simultaneously parsing the string into segments.

6.2.1 *Character Appearance*

Each segment is scored for a particular character and segment width $|s|$ by a learned linear discriminant:

$$U^A(s, y, \mathbf{x}; \boldsymbol{\theta}^A) = |s|\boldsymbol{\theta}^A(|s|, y)^\top F(s, \mathbf{x}). \tag{7}$$

The linear parameters $\boldsymbol{\theta}^A$ of the inner product are dependent not only on the character identity y, but also the size of the segment, $|s| = t - r$. Therefore U^A learns both the appearance and width of characters. The discriminative nature of the model allows us to use image features outside the segment without violating any independence assumptions. Greater image context could help disambiguate some characters.

The probability (6) could be biased by the number of segments. This potential bias is handled automatically by an appropriate learning process that can give different segment widths different score magnitudes. However, our simplified piecewise training strategy (described below) prevents this beneficial side effect. Therefore, we include the simple width multiplier $|s|$ in the appearance energy (7), which effectively assigns the inner product to every index along the segment.

We resize the normalized image to a 25 px font height; the word box height is therefore either 12.5 px (assuming the upper guide is the mean line) or 18 px (if the cap line). For efficiency, we quantize the valid segment widths to be $\mathcal{W} = \{4, 8, 12, 16, 20, 24, 32\}$ pixels. Using a 32×32 window centered in the segment s, the features $F(s, \mathbf{x})$ are complex magnitudes of steerable pyramid filters with eight orientations and one scale (omitting the low and high pass bands) [49]. For RGB images, we take the largest filter response among the three color channels at each pixel. Contrast normalization is critical to performance. We divide each filter response by the ℓ_2 norm of all responses in an 8×16 window (efficiently computed with box filters), clip at 0.2, and renormalize. We also include the binary image as a feature because it increases accuracy by anchoring the phase invariant steerable pyramid feature magnitudes. Thus, $F(s, \mathbf{x})$ is a $32 \times 32 \times 9 = 9{,}216$ element vector, plus an added bias dimension.

6.2.2 *Language: Character Bigrams and a Lexicon*

Linguistic properties provide helpful top-down cues for character recognition in challenging images. The character N-gram is a ubiquitous feature for OCR, STR, and handwriting recognition. In this model, each pair of neighboring character segments s', s with labels y', y has a bigram score,

$$U^B(s', s, y', y; \boldsymbol{\theta}^B) = (t - r')\theta^B(y', y), \tag{8}$$

with the first character segment $s' = (r', t')$ and the subsequent segment $s = (r, t)$. When bigram scores are not a completely integrated part of the model, using them can introduce problematic biases (i.e., for longer or shorter strings) that must somehow be rectified. Wang et al. [50] examine several possibilities. Just as in the character appearances, we could make the bigram scores conditional not only on the labels, but on the segment lengths, i.e., $\theta^B(s's, y', y)$ [51]. Instead, we compensate for training approximations by factoring segment width into a multiplier, rather than a conditioning term, in effect tying the bigram parameters across all segment widths. Note that two bigram scores get counted on every segment except the first and last; our implementation corrects for this bias.

To facilitate a soft preference for character sequences that compose a lexicon word, we use a different energy,

$$U^L(s', s; \theta^L) = (t - r')\theta^L, \tag{9}$$

in lieu of the bigram score for segments belonging to a lexicon word. See Section 6.3.2 below for computational ramifications.

6.2.3 *Geometry: Character Gap or Overlap*

A pair of neighboring segments may either overlap or have a gap between them. We allow character bounding boxes to overlap (as in italics or an fi ligature), with a quadratic penalty:

$$U^O(s', s; \boldsymbol{\theta}^O) = \begin{cases} 0, & r - t' < \varepsilon, \\ \infty, & r - t' > \tau, \\ (r - t')^2\theta_2^O + (r - t')\theta_1^O, & \text{otherwise.} \end{cases} \tag{10}$$

Because character widths are quantized, we allow overlap that could arise from quantization error $\varepsilon = 2$ without penalty and a hard limit of up to half the segment width, $\tau = \min\{\frac{1}{2}\min\{|s'|, |s|\}, 8\}$. Though we manually fixed $(\theta_2^O, \theta_1^O) = (0.0234, 0.7416)$ in our experiments, these are easily learned.

When a pair of neighboring character segments have a gap between them, the gap is scored by a learned compatibility function,

$$U^G(s', s, \mathbf{x}; \boldsymbol{\theta}^G) = \sum_{i=t'+1}^{r-1} (\boldsymbol{\theta}^G)^\top F(i, \mathbf{x}), \tag{11}$$

which is a sum of the scores for each index (column) i in the gap between segments s' and s. The same image features of the character appearance model are multiplied by a different set of learned linear parameters. In practice, we limit the gap to 12 pixels.

6.3 Model Inference

The recognition task is to find the most probable segmentation and labeling $(\hat{\mathbf{s}}, \hat{\mathbf{y}}) = \arg\max_{\mathbf{s},\mathbf{y}} p(\mathbf{s}, \mathbf{y} \mid \mathbf{x}, \boldsymbol{\theta})$. Equivalently, we minimize the total of the energies described in the previous section. This inference process may be done in a model with or without a lexicon. For simplicity, we begin by describing a lexicon-free model.

6.3.1 *Lexicon-Free Parsing*

We build up a 2D dynamic programming table to find the optimal parse. Let $S(t, y)$ be the optimal score for a segment ending at index t with character y. Letting $t = 1$ be the first column index of the sequence, we iteratively build the table via the recurrence relation

$$S(t, y) = \begin{cases} \max_{t', r, y'} S(t', y') - P(t', r, t, y', y, 0), & t > 0, \\ 0, & t = 0, \\ -\infty, & t < 0, \end{cases} \tag{12}$$

where $P(t', r, t, y', y, 0)$ represents the additional parse score for adding a segment $s = (r, t)$ with character label y, while the previous character y' ended at t'. The last argument 0 indicates that the additional character is not forming part of a lexicon word.

Energy terms detailed in the previous section compose the additional parse score:

$$\begin{aligned} P(t', r, t, y', y, w) = U^A(s, y) + wU^L(s', s) \\ + (1 - w)U^B(s', s, y', y) \\ + U^O(s', s) + U^G(s', s), \end{aligned} \tag{13}$$

where $s = (r, t)$ and $s' = (r', t')$. The total score for a given segmentation and labeling is the negated sum of all the P terms, which of course is the value inside the exponential of the corresponding probability model (6).

We trace back the optimal parse by storing the argmax values of t', r, and y' for each step t. Here we especially note that no separate word segmentation must be done; it is completely automatic because a space is among the characters to be recognized.

6.3.2 Lexicon-Based Parsing

When incorporating a lexicon (and thereby allowing a bias for strings to be recognized as lexicon words), the optimization becomes significantly more complex. If the text region is assumed to be a single word, we simply find the highest-scoring word among parses constrained to the lexicon. The dynamic programming can be interleaved with a trie traversal, reusing optimal parse information from the shared prefix substrings to speed processing [28], [52], [47].

In general, we cannot assume detected text lines contain just one word; words may start or end at any point in the line. To handle this generalization, we maintain two parallel dynamic programming tables, one for nonlexicon parses and another with lexicon restrictions.

As before, the table $S(t, y)$ corresponds to the best parse for an arbitrary character y ending at point t. It is built by adding optimal segments and characters using the parse score (13). A new table $W(t)$ corresponds to the best parse for any lexicon word ending at t. The connection between these two tables occurs when $y = \sqcup$ is a space. The optimal character given by $S(t, y)$ may have been preceded by a lexicon word; conversely, the optimal *word* calculated by $W(t)$ may have been preceded by a nonlexical character. Mutually dependent recurrence relations link these two tables.

With a lexicon, the primary recurrence for $S(t, y)$ now has two cases. When $y \neq \sqcup$, the parse score is the sum of the best previous score plus the score for adding the new character segment, as given by (12) earlier. When the new $y = \sqcup$ is a space, the end of a character string is signaled, and that string may either be a lexicon word or not. In this case, the dynamic programming must determine whether the optimal parse is to accept the previous lexicon word S_ℓ or to take the nonlexicon parse ending before the space $S_{\bar{\ell}}$:

$$S(t, \sqcup) = \max\{S_\ell(t), S_{\bar{\ell}}(t)\}, \tag{14}$$

$$S_\ell(t) = \max_{t', r} W(t') - P(t', r, t, \widehat{y}_{t'}, \sqcup, 0), \tag{15}$$

$$S_{\bar{\ell}}(t) = \max_{t', r, y'} S(t', y') - P(t', r, t, y', \sqcup, 0), \tag{16}$$

where $\widehat{y}_{t'}$ is the last character of the word parse ending at t' and all other terms are as before.

Building the table $W(t)$ for the best score of a lexicon word ending at t requires a new recurrence $C(t'', t', \mathbf{y})$, which scores the best *partial* lexicon word $\mathbf{y}$ over the span from t'' to t'. The score for extending the subword by the character y, ending at index t is

$$C(t'', t, [\mathbf{y} \mid y]) = \max_{t', r} C(t'', t', \mathbf{y}) - P(t', r, t, y', y, 1), \tag{17}$$

where y' is the last character of $\mathbf{y}$ and $[\mathbf{y} \mid y]$ forms a new partial lexicon word. As before, the term P is a score for a parse of a particular character including the appearance and geometry scores. However, the language model is altered because the character is now part of a (hypothesized) lexicon word. The score U^L for constituents of a lexicon word thus replaces the bigram score U^B entirely, as indicated by the last argument, $w = 1$. The lexical recurrence relation $W(t)$ is similar to the original table $S(t, y)$, but with a larger spatial scale:

$$W(t) = \max_{t''} S(t'', \sqcup) + \max_{\mathbf{y} \in \mathcal{L}} C(t'', t, \mathbf{y}), \tag{18}$$

where $\mathcal{L}$ is the set of complete lexicon words. The score $W(t)$ builds upon the previous scores S, but adds an additional term C that represents the optimal score for *any* lexicon word following a space.

6.3.3 Computational Complexity and Beam Search

This approach begets a problem of scale. Although the complexity is linear in the number of lexicon words and the length of the image to be parsed, our proposed method maintains hypotheses for every possible word beginning at every possible location. This cross product of possibilities is impossibly slow in practice, so approximations must be introduced.

For the span from t'' to t, the function $C(t'', t, \mathbf{y})$ ranks subwords $\mathbf{y}$, most of which are extremely unlikely. At the risk of error, we eliminate initially low-scoring candidates for each span to speed processing. Every subword eliminated further eliminates words building upon that subword parse from consideration. With fewer calculations of the subword score (17) to be made, there are fewer complete words to score. Such pruning is a form of beam search common to dynamic programming algorithms for speech recognition. While several pruning strategies exist, our earlier work indicated that a simple N-best list was sufficient for this task [23]. Under this standard strategy for Viterbi beam search, we simply sort the scores C for a given region (t'' to t), keeping the N best subword parses. We use $N = 50$ for the small-vocabulary SVT data and $N = 25$ for the others.

With our unoptimized Java implementation, the average parse runtime for the ICDAR data on a standard desktop machine is 40 sec/word with a lexicon and 20 sec/word without. SVT word spotting is 25 sec/word. We note that significant opportunities for parallelism abound, for example, by computing maxima over nearly 3,000 values for each nonlexical parse score $S(t, y)$ alone.

6.4 Model Training

The discriminative semi-Markov model (6) has a learning process analogous to conventional probabilistic models: maximizing a regularized version of the convex conditional log likelihood [24]. Unlike the information extraction tasks for which the model was first proposed, such a learning scheme poses two major drawbacks on this task. First and foremost, such a scheme requires the training data be like the testing data in that both labeling and segmentation (i.e., context) are required. This restriction is problematic because tens or even hundreds of thousands of examples are required for effective character recognition [53], yet the largest STR data sets only have a few thousand characters in context. The second major drawback is time, because the model performs segmentation along with recognition over several iterations of gradient descent.

The many works that use weighted FSTs sidestep these problems by simply learning modules (e.g., character recognition, N-grams) separately, perhaps adjusting the relative module weights in an ad hoc fashion [13], [17], [16]. We formalize this approach under the piecewise pseudolikelihood interpretation of Sutton and McCallum [54]. Our two-stage training process first temporarily decouples the appearance and bigram modules to reduce train time and increase available training data. First, piecewise training maximizes a lower bound on the likelihood of the training data by learning the energy functions separately. Because all the energies are linear in the parameters $\boldsymbol{\theta}$, the original and piecewise log likelihoods are convex and optimized with a limited-memory second-order Newton method for gradient descent. In the second training stage, we recouple all the energies for an important lower-dimensional scale tuning via the generalized perceptron loss [55].

First, the appearance and gap parameters, $\boldsymbol{\theta}^A$ and $\boldsymbol{\theta}^G$, are trained on a set of rendered examples (see Weinman's dissertation for details [23, chs. 5 and 6]) from 1,866 fonts given random contrast, bias, noise, borders, and neighboring characters. To mimic the wide variety of scene text, we include versions of these fonts horizontally scaled at $2^{-1/2}$, $2^{-1/4}$, and $2^{1/2}$ for a total of $1{,}866 \times 62 \times 4 = 462{,}768$ characters plus 16,794 spaces and gaps. The quantized horizontal range of each raw character image is taken as the segment width for learning $\boldsymbol{\theta}^A(|s|, y)$. For efficiency, only segmentations of the quantized widths centered in the training window are considered during learning.

Bigram parameters $\boldsymbol{\theta}^B$ are simultaneously trained in a piecewise fashion on a text corpus [20]. We collapse digits into a single class because they occur infrequently and exhibit no meaningful sequence patterns. Conversely, all-uppercase letter signs occur frequently, so intracase transition parameters are tied across case, for example, $\theta^B(\mathtt{T},\mathtt{H}) = \theta^B(\mathtt{t},\mathtt{h})$; intercase transitions are preserved so that in general $\theta^B(\mathtt{T},\mathtt{h}) \neq \theta^B(\mathtt{t},\mathtt{h}) \neq \theta^B(\mathtt{t},\mathtt{H})$. Once $\boldsymbol{\theta}^B$ is learned, we set θ^L to twice the median bigram weight for all bigrams over words in our lexicon $\mathcal{L}$, a value which is tuned in the secondary stage below.

Both $\boldsymbol{\theta}^A$ and $\boldsymbol{\theta}^B$ are trained with a Laplace prior (ℓ_1 regularization). We choose the weight of the ℓ_1 penalty on $\boldsymbol{\theta}^B$ with a tenfold cross-validation, optimizing the total likelihood of all held out folds. The ℓ_1 penalty on $\boldsymbol{\theta}^A$ is instead chosen by minimizing recognition error on a training validation set after the entire model has been learned. Delayed validation helps compensate for piecewise training with fixed segments.

TAlthough piecewise training maximizes a lower bound on the true likelihood, there is significant risk that the scales of the energy functions learned in this fashion are not commensurate. Therefore, we undertake a simple secondary training step for positive weights $\boldsymbol{\lambda} = (\lambda^A, \lambda^B, \lambda^L, \lambda^G, \lambda^O)$ to adjust the scale of all five energy components—appearance $\lambda^A \cdot U^A$, bigrams $\lambda^B \cdot U^B$, lexicon $\lambda^L \cdot U^L$, gap $\lambda^G \cdot U^G$, and overlap $\lambda^O \cdot U^O$—by using gradient descent to minimize the perceptron loss

$$E(\boldsymbol{\lambda}) = \sum_i \min_{\mathbf{s}} U(\mathbf{s}, \mathbf{y}_i, \mathbf{x}_i; \boldsymbol{\lambda}, \boldsymbol{\theta}) - \min_{\mathbf{s}', \mathbf{y}'} U(\mathbf{s}', \mathbf{y}', \mathbf{x}_i; \boldsymbol{\lambda}, \boldsymbol{\theta}), \tag{19}$$

where U is the total energy of the probability model (6) whose components are scaled by $\boldsymbol{\lambda}$, $\mathbf{y}_i$ is a ground truth character labeling, and $\boldsymbol{\theta}$ is fixed at the values learned by piecewise training. $E(\boldsymbol{\lambda})$ has a lower bound at zero where the best interpretation $\mathbf{y}'$ matches the ground truth [55]. In practice, pretraining $\boldsymbol{\theta}$ prevents the perceptron functional from collapsing U to be zero everywhere. Unlike the first piecewise training stage, we minimize the perceptron loss (19) on contextual training data that is like the testing data, where full segmentation optimization is required.

TABLE 3
Word Spotting Results on the Street View Text Data (647 Words, 249 Images) [13]

Method	Word Error (%)
ABBYY [19]	65
Wang *et al.* [19]	43
Mishra *et al.* [17]	26.74
Proposed Method	21.95

7 Experiments

We evaluate our system on three data sets that each test slightly different and increasingly difficult problems: a small closed-vocabulary word spotting task, an open-vocabulary word recognition task, and two open-vocabulary word segmentation and recognition tasks, one with noisy detections, giving complete end-to-end performance.

7.1 Closed-Vocabulary Word Spotting

Wang and Belongie's street view text data set features 647 test words in 249 images, each with an associated lexicon of about 50 words [19]. Annotated bounding boxes are drawn from this small, closed vocabulary. The evaluation is case insensitive and the lexicon is given in all uppercase letters. To perform this word spotting task, our recognition model simply ignores letter case, prohibiting spaces and nonlexical parses. Table 3 shows our method to be the state of the art.

7.2 Open-Vocabulary Word Recognition

The 2011 ICDAR robust reading competition [18] features 1,189 word bounding boxes over 255 scene images, including words from several languages. Though no lexicon is given, it is such a strong source of information that we create one (using SCOWL,[1] with British and American English) including proper names, abbreviations, and contractions (with all punctuation removed). Our case-sensitive model ignores word frequency, so we only include words up to the 50th frequency percentile and add uppercase and leading caps versions of all words to the lexicon (along with the training words) so that $|\mathcal{L}| = 244{,}033$ words, including 884 (74 percent) of the test words. Table 4 compares results from the recent competition. Character/word error is the average of word edit distance divided by word length. The

1. http://wordlist.sourceforge.net, revision 6.

TABLE 4
Word Recognition Results on the ICDAR 2011 Scenes Data (6,393 Characters, 1,189 Words, and 255 Images) [18]

	Error (%)		
Method	Character/Word	Character	Word
Neumann *et al.* [18]	36.14	-	66.89
Lee *et al.* [18]	26.78	-	64.4
Yang *et al.* [18]	14.82	-	58.8
Appearance + Geometry	44.00	38.59	60.72
+ Bigram	40.00	34.21	54.50
+ Lexicon	36.24	29.36	42.30

TABLE 5
Word Segmentation and Recognition Results on the VIDI Data (1,164 Characters, 192 Words, and 86 Images) [20]

	Error (%)	
Model Features	Character	Word
Appearance + Geometry	16.84	58.33
+ Bigrams	12.89	43.23
+ Lexicon	10.65	24.48

TABLE 6
Average End-to-End Results for ICDAR 2011 Scenes

Method and Task	Prec.	Recall	F1
Neumann & Matas [6] Detection	73.1	64.7	68.7
Neumann & Matas [6] Recognition	37.1	37.2	36.5
Yi & Tian [56] Detection	67.22	58.09	62.32
Proposed Method Recognition	41.10	36.45	33.71
Omitting One-Character Words	46.60	36.27	36.40

standard character error divides the total edit distance (over all words) by the total number of characters in the data. Case-sensitive evaluations (see Table 4) show our method to be the state of the art for this task as well.

7.3 Word Segmentation and Recognition

TTo test our model's integrated word segmentation capabilities, we first use a punctuation-free subset of the VIDI data by Weinman et al. [20], which provides approximate text baselines and a prenormalized font size of 100 px for 1,164 characters in 192 words from 86 images. Words must be segmented automatically because the problem input is a line of text. We measure word error (including all stop words) with the ISRI OCR performance toolkit program `wordacc`,[2] modified to include letter case sensitivity and number strings, which the original ignores. Table 5 shows quantitative results.

For a complete end-to-end test including detection, normalization, word segmentation, and recognition, we use text detection results of Yi and Tian [56] as submitted to the 2011 ICDAR detection competition [18], where it placed second. Fig. 12 shows some examples, while Table 6 reports detection results with the 2011 competition methodology [18] and recognition results using the case-sensitive methodology of Lucas et al. [7]. Many spurious detections give only one-character words, so we report results where these are dropped from the system output; ground truth is unaltered.

7.4 Discussion

We eliminate nearly 18 percent of errors on the SVT word spotting task. The task is greatly simplified by the small lexicon, and these results illustrate the relative difficulty of

2. http://code.google.com/p/isri-ocr-evaluation-tools.

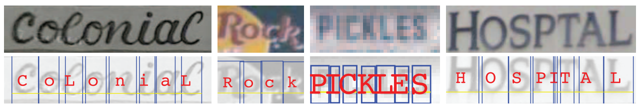

Fig. 9. Street view text word spotting. Evaluation is case insensitive, but recognized lettercase is shown.

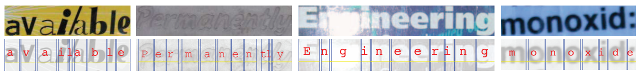

Fig. 10. ICDAR 2011 word recognition.

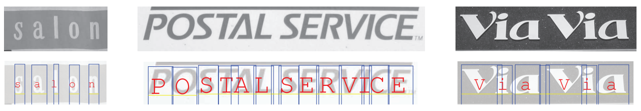

Fig. 11. VIDI word segmentation and recognition.

the remaining incorrect words. In the open-vocabulary word recognition setting, we significantly improve state of the art ICDAR 2011 results. We note word recognition error rises 8 percent when regions are not grouped into lines for guide fitting, indicating this preprocessing step gives a significant performance boost.

The comparatively high character/word error on the ICDAR data indicates many failures happen earlier in the recognition pipeline, leaving little hope of recognizing even a portion of the word. Our model must also infer whether the top guide is the mean or capline. If the objectively better interpretation were chosen (that with lesser error), the overall character error would be reduced by nearly a third and the word error by 14 percent. Resolving more of these common error cases correctly would bring modest improvement.

In the open vocabulary VIDI data, we demonstrate the utility of joint word segmentation and recognition. Note the large intraword gaps and small interword spaces correctly identified in Fig. 11. Overall error rates are lower than for ICDAR and SVT because the guidelines are given; yet some fonts and complex backgrounds make the remaining errors challenging.

On the end-to-end task, top-down word segmentation resolves cases where the bottom-up detector cannot reliably segment words. Despite using lower quality text detections as input, we achieve results at par with the state of the art. A better initial text detection score would likely yield even better overall results (see Section 7.2).

TThe lexicon notably reduces word recognition errors, by 22 percent on the ICDAR word recognition task and 43 percent on the VIDI task, which includes word segmentation. However, it can be a double-edged sword. A mosaicing error (see Fig. 9, right) requires hallucinating the "I" character for word spotting, a strategy that can backfire in an open vocabulary setting (see Fig. 10, right).

8 Conclusion

We have presented a system that handles many stages of scene text reading in a probabilistic fashion, from binarization to appearance normalization to character segmentation and recognition. We do not rely on individual character detections, which are prone to false negatives. Only a coarse binarization is necessary for detecting guidelines that allow us to handle curved and rotated text. Our model uses a

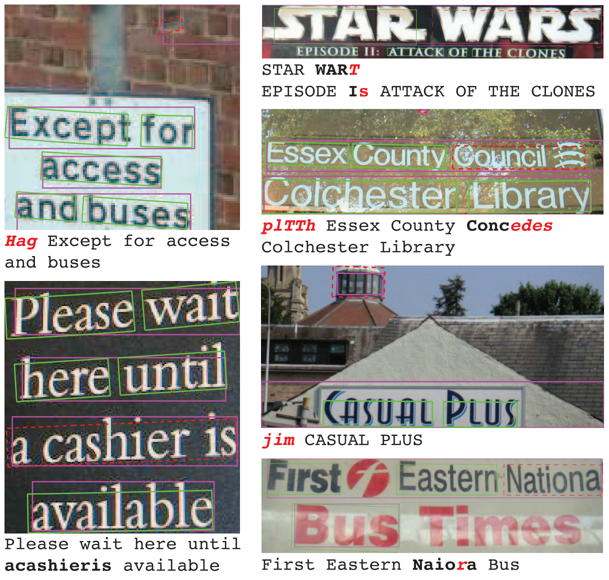
Hag Except for access and buses
STAR **WAR*T***
EPISODE **I*s*** ATTACK OF THE CLONES
p1TTh Essex County **Conc*edes***
Colchester Library
Please wait here until **acashieris** available
jim CASUAL PLUS
First Eastern **Naio*ra*** Bus

Fig. 12. End-to-end examples: input detections (by Yi and Tian [56]) shown in magenta, correctly segmented/recognized words in green, and incorrect regions in dashed red. Incorrect words printed in bold, with incorrect characters in italicized red.

hybrid open/closed vocabulary approach to balance bottom-up signals with top-down priors. Most importantly, it fully integrates segmentation and recognition.

While these latter two stages are integrated, partially in training and fully in testing, we should like for the entire system to be even more integrated, in learning first [26], but more so that later pipeline stages can provide feedback to earlier modules that may refine grouping, binarization, or guideline hypotheses in an iterative fashion, as in human information processing [57]. Given the difficulty of bottom-up processing, further opportunities for passing multiple hypotheses from one stage to the next are worthy of exploration, though care must be taken to avoid a combinatorial rise in complexity.

The binarization, guideline fitting, and character classification modules remain somewhat rudimentary. It is the integration of these stages and evaluating multiple hypotheses that makes the system perform well. However, further refinements to these key modules could bring even better performance.

Scene text recognition is difficult—the world's vivid colors, uncontrolled lighting, and unpredictable perspectives conspire to make general machine reading a "grand challenge"-worthy task. Like other recent challenges, with additional concentrated engineering and scientific efforts, we believe STR can be solved.

Acknowledgments

The authors thank N. Howe, J. Jonkman, E. Learned-Miller, and C. Sutton for helpful conversations, the anonymous reviewers for feedback that improved the paper, A. Mishra for providing binarized word images, and C. Yi for the text detections. Messrs. Weinman, Butler, and Knoll gratefully acknowledge financial support from Grinnell College. Ms. Feild was supported by a US National Science Foundation Graduate Research Fellowship.

References

[1] R. Darnton, "The Library in the New Age," *The New York Rev. of Books*, vol. 55, June 2008.
[2] V. Wu, R. Manmatha, and E.M. Riseman, "Textfinder: An Automatic System to Detect and Recognize Text in Images," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 21, no. 11, pp. 1224-1229, Nov. 1999.
[3] X. Chen, J. Yang, J. Zhang, and A. Waibel, "Automatic Detection and Recognition of Signs from Natural Scenes," *IEEE Trans. Image Processing*, vol. 13, no. 1, pp. 87-99, Jan. 2004.
[4] X. Chen and A.L. Yuille, "Detecting and Reading Text in Natural Scenes," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 366-373, 2004.
[5] B. Epshtein, E. Ofek, and Y. Wexler, "Detecting Text in Natural Scenes with Stroke Width Transform," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 2963-2970, 2010.
[6] L. Neumann and J. Matas, "Real-Time Scene Text Localization and Recognition," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 3538-3545, 2012.
[7] S.M. Lucas, A. Panaretos, L. Sosa, A. Tang, S. Wong, and R. Young, "ICDAR 2003 Robust Reading Competitions," *Proc. Int'l Conf. Document Analysis and Recognition*, vol. 2, pp. 682-687, 2003.
[8] J. Ohya, A. Shio, and S. Akamatsu, "Recognizing Characters in Scene Images," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 16, no. 2, pp. 214-220, Feb. 1994.
[9] C. Thillou, S. Ferreira, and B. Gosselin, "An Embedded Application for Degraded Text Recognition," *Eurasip J. Applied Signal Processing*, vol. 13, pp. 2127-2135, 2005.
[10] J.J. Weinman and E. Learned-Miller, "Improving Recognition of Novel Input with Similarity," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 308-315, June 2006.
[11] S.M. Lucas, "Text Locating Competition Results," *Proc. Int'l Conf. Document Analysis and Recognition*, pp. 80-85, 2005.
[12] Z. Saidane, C. Garcia, and J.L. Dugelay, "The Image Text Recognition Graph (iTRG)," *Proc. Int'l Conf. Multimedia and Expo*, pp. 266-269, 2009.
[13] K. Wang and S. Belongie, "Word Spotting in the Wild," *Proc. European Conf. Computer Vision*, Sept. 2010.
[14] L. Neumann and J. Matas, "A Method for Text Localization and Recognition in Real-World Images," *Proc. Asian Conf. Computer Vision*, pp. 770-783, 2010.
[15] T. Yamazoe, M. Etoh, T. Yoshimura, and K. Tsujino, "Hypothesis Preservation Approach to Scene Text Recognition with Weighted Finite-State Transducer," *Proc. Int'l Conf. Document Analysis and Recognition*, pp. 359-363, 2011.
[16] K. Elagouni, C. Garcia, F. Mamalet, and P. Sébillot, "Combining Multi-Scale Character Recognition and Linguistic Knowledge for Natural Scene Text OCR," *Proc. Int'l Workshop Document Analysis Systems*, pp. 120-124, Mar. 2012.
[17] A. Mishra, K. Alahari, and C.V. Jawahar, "Top-Down and Bottom-Up Cues for Scene Text Recognition," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 2687-2694, 2012.
[18] A. Shahab, F. Shafait, and A. Dengel, "ICDAR 2011 Robust Reading Competition Challenge 2: Reading Text in Scene Images," *Proc. Int'l Conf. Document Analysis and Recognition*, pp. 1491-1496, 2011.
[19] K. Wang, B. Babenko, and S. Belongie, "End-to-End Scene Text Recognition," *Proc. IEEE Int'l Conf. Computer Vision*, pp. 1457-1464, 2011.
[20] J.J. Weinman, E. Learned-Miller, and A. Hanson, "Scene Text Recognition Using Similarity and a Lexicon with Sparse Belief Propagation," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 31, no. 10, pp. 1733-1746, Oct. 2009.
[21] P. Shivakumara, S. Bhowmick, B. Su, C.L. Tan, and U. Pal, "A New Gradient Based Character Segmentation Method for Video Text Recognition," *Proc. Int'l Conf. Document Analysis and Recognition*, pp. 126-130, 2011.
[22] J.J. Weinman, E. Learned-Miller, and A. Hanson, "A Discriminative Semi-Markov Model for Robust Scene Text Recognition," *Proc. Int'l Conf. Pattern Recognition*, Dec. 2008.
[23] J.J. Weinman, "Unified Detection and Recognition for Reading Text in Scene Images," PhD thesis, Univ. of Massachusetts Amherst, 2008.
[24] S. Sarawagi and W.W. Cohen, "Semi-Markov Conditional Random Fields for Information Extraction," *Advances in Neural Information Processing Systems*, pp. 1185-1192, MIT Press, 2005.

[25] J.D. Ferguson, "Variable Duration Models for Speech," *Proc. Symp. Application of Hidden Markov Models to Text and Speech,* pp. 143-179, 1980.

[26] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-Based Learning Applied to Document Recognition," *Proc. IEEE,* vol. 86, no. 11, pp. 2278-2324, Nov. 1998.

[27] X. Huang, A. Acero, and H.-W. Hon, *Spoken Language Processing: A Guide to Theory, Algorithm, and System Development.* Prentice Hall PTR, 2001.

[28] C.-L. Liu, M. Koga, and H. Fujisawa, "Lexicon-Driven Segmentation and Recognition of Handwritten Character Strings for Japanese Address Reading," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 24, no. 11, pp. 1425-1437, Nov. 2002.

[29] H.I. Koo and N.I. Cho, "Text-Line Extraction in Handwritten Chinese Documents Based on an Energy Minimization Framework," *IEEE Trans. Image Processing,* vol. 21, no. 3, pp. 1169-1175, Mar. 2012.

[30] Y.-F. Pan, X. Hou, and C.-L. Liu, "Text Localization in Natural Scene Images Based on Conditional Random Field," *Proc. Int'l Conf. Document Analysis and Recognition,* pp. 6-10, 2009.

[31] H. Zhang, C. Liu, C. Yang, X. Ding, and K. Wang, "An Improved Scene Text Extraction Method Using Conditional Random Field and Optical Character Recognition," *Proc. Int'l Conf. Document Analysis and Recognition,* pp. 708-712, 2011.

[32] J. Zhang and R. Kasturi, "Character Energy and Link Energy-Based Text Extraction in Scene Images," *Proc. Asian Conf. Computer Vision,* pp. 308-320, 2010.

[33] K. Kita and T. Wakahara, "Binarization of Color Characters in Scene Images Using K-Means Clustering and Support Vector Machines," *Proc. Int'l Conf. Pattern Recognition,* pp. 3183-3186, 2010.

[34] C. Zeng, W. Jia, and X. He, "An Algorithm for Colour-Based Natural Scene Text Segmentation," *Proc. Fourth Int'l Workshop Camera-Based Document Analysis and Recognition,* pp. 58-68, 2012.

[35] M.S. Cho, J.-H. Seok, S. Lee, and J.H. Kim, "Scene Text Extraction by Superpixel CRFs Combining Multiple Character Features," *Proc. Int'l Conf. Document Analysis and Recognition,* pp. 1034-1038, 2011.

[36] A. Mishra, K. Alahari, and C. Jawahar, "An MRF Model for Binarization of Natural Scene Text," *Proc. Int'l Conf. Document Analysis and Recognition,* pp. 11-16, 2011.

[37] Z. Tu and S.-C. Zhu, "Image Segmentation by Data-Driven Markov Chain Monte Carlo," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 24, no. 5, pp. 657-673, May 2002.

[38] J. Feild and E. Learned Miller, "Scene Text Recognition with Bilateral Regression," Technical Report UM-CS-2012-021, Univ. of Massachusetts-Amherst, Computer Science Research Center, Univ. of Massachusetts, 2012.

[39] M.-K. Hu, "Visual Pattern Recognition by Moment Invariants," *IRE Trans. Information Theory,* vol. 8, no. 2, pp. 179-187, Feb. 1962.

[40] P.M. Williams, "Bayesian Regularization and Pruning Using a Laplace Prior," *Neural Computation,* vol. 7, pp. 117-143, 1995.

[41] J. Jung, S. Lee, M. Cho, and J. Kim, "Touch TT: Scene Text Extractor Using Touchscreen Interface," *ETRI J.,* vol. 33, no. 1, pp. 78-88, 2011.

[42] J.J. Hull, *Document Image Skew Detection: Survey and Annotated Bibliography,* vol. Document Analysis Systems II of Series in Machine Perception and Artificial Intelligence, pp. 40-64, 1998.

[43] Y. Bengio and Y.L. Cun, "Word Normalization for On-Line Handwritten Word Recognition," *Proc. Int'l Conf. Pattern Recognition,* pp. 409-413, 1994.

[44] T. Caesar, J.M. Gloger, and E. Mandler, "Estimating the Baseline for Written Material," *Proc. Int'l Conf. Document Analysis and Recognition,* vol. 1, pp. 382-385, Aug. 1995.

[45] L. Neumann and J. Matas, "Text Localization in Real-World Images Using Efficiently Pruned Exhaustive Search," *Proc. Int'l Conf. Document Analysis and Recognition,* pp. 687-691, 2011.

[46] D.J. MacKay, "Bayesian Interpolation," *Neural Computation,* vol. 4, pp. 415-448, 1992.

[47] C. Jacobs, P.Y. Simard, P. Viola, and J. Rinker, "Text Recognition of Low-Resolution Document Images," *Proc. Int'l Conf. Document Analysis and Recognition,* pp. 695-699, 2005.

[48] M.-Y. Chen, A. Kundu, and S.N. Srihari, "Variable Duration Hidden Markov Model and Morphological Segmentation for Handwritten Word Recognition," *IEEE Trans. Image Processing,* vol. 4, no. 12, pp. 1675-1688, Dec. 1995.

[49] E.P. Simoncelli and W.T. Freeman, "The Steerable Pyramid: A Flexible Architecture for Multi-Scale Derivative Computation," *Proc. Int'l Conf. Image Processing,* vol. 3, pp. 444-447, 1995.

[50] Q.-F. Wang, F. Yin, and C.-L. Liu, "Handwritten Chinese Text Recognition by Integrating Multiple Contexts," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 34, no. 8, pp. 1469-1481, Aug. 2012.

[51] J.J. Weinman, "Typographical Features for Scene Text Recognition," *Proc. Int'l Conf. Pattern Recognition,* pp. 3987-3990, 2010.

[52] S.M. Lucas, G. Patoulas, and A.C. Downton, "Fast Lexicon-Based Word Recognition in Noisy Index Card Images," *Proc. Int'l Conf. Document Analysis and Recognition,* vol. 1, pp. 462-466, 2003.

[53] P.Y. Simard, D. Steinkraus, and J.C. Platt, "Best Practices for Convolutional Neural Networks Applied to Visual Document Analysis," *Proc. Int'l Conf. Document Analysis and Recognition,* vol. 2, pp. 958-963, 2003.

[54] C. Sutton and A. McCallum, "Piecewise Training for Structured Prediction," *Machine Learning,* vol. 77, nos. 2/3, pp. 165-194, 2009.

[55] Y. LeCun, S. Chopra, R. Hadsell, M. Ranzato, and F. Huang, "A Tutorial on Energy-Based Learning," *Predicting Structured Data,* G. Bakir, T. Hofman, B. Schölkopf, A. Smola, and B. Taskar, eds., MIT Press, 2006.

[56] C. Yi and Y. Tian, "Text String Detection from Natural Scenes by Structure-Based Partition and Grouping," *IEEE Trans. Image Processing,* vol. 20, no. 9, pp. 2594-2605, Sept. 2011.

[57] J.L. McClelland and D.E. Rumelhart, "An Interactive Activation Model of Context Effects in Letter Perception: Part 1. An Account of Basic Findings," *Psychological Rev.,* vol. 88, pp. 375-407, Sept. 1981.

Jerod J. Weinman received the BS degree in computer science and mathematics from Rose-Hulman Institute of Technology in 2001, and the MSc and PhD degrees in computer science from the University of Massachusetts Amherst in 2005 and 2008, respectively. He is an assistant professor at Grinnell College, with research interests in computer vision and machine learning. He is a member of the IEEE, the ACM, ACM SIGCAS, and ACM SIGCSE.

Zachary Butler received the BA degree in computer science and mathematics from Grinnell College in 2013. He is currently working toward the PhD degree at the University of California, Irvine. His research interests include the intersection of machine learning and computer vision.

Dugan Knoll received the BA degree in computer science from Grinnell College in 2012. He is currently a web developer at Salem Health, Oregon, and plans to attend a graduate program in computer science in the future.

Jacqueline Feild received the BS degree from Loyola University Maryland in 2007 and the MS degree from the University of Massachusetts Amherst in 2010. She is working toward the PhD degree in the School of Computer Science at the University of Massachusetts Amherst. She works in the Computer Vision Lab with Dr. Erik Learned-Miller.