Large-Lexicon Attribute-Consistent Text Recognition in Natural Images

Tatiana Novikova[1], Olga Barinova[1], Pushmeet Kohli[2], Victor Lempitsky[3]

[1]Lomonosov Moscow State University, [2]Microsoft Research Cambridge, [3]Yandex

Abstract. This paper proposes a new model for the task of word recognition in natural images that simultaneously models visual and lexicon consistency of words in a single probabilistic model. Our approach combines local likelihood and pairwise positional consistency priors with higher order priors that enforce consistency of characters (lexicon) and their attributes (font and colour). Unlike traditional stage-based methods, word recognition in our framework is performed by estimating the maximum a posteriori (MAP) solution under the joint posterior distribution of the model. MAP inference in our model is performed through the use of weighted finite-state transducers (WFSTs). We show how the efficiency of certain operations on WFSTs can be utilized to find the most likely word under the model in an efficient manner. We evaluate our method on a range of challenging datasets (ICDAR'03, SVT, ICDAR'11). Experimental results demonstrate that our method outperforms state-of-the-art methods for cropped word recognition.

1 Introduction

Recent years have seen a surge of interest in the task of text understanding in "natural" photographs. This has primarily been driven by the large number of potential applications including those concerned with mobile vision and robotics. In general, in contrast to the understanding of text in scanned documents (which is a mature technology), understanding text in natural photographs of man-made environments remains a hard problem and a topic of an active research in the vision community [1], [2], [3], [4]. So far, most of these efforts have been pointed at adapting low-level and mid-level computer vision frameworks, e.g. morphological analysis [5], belief propagation in second-order random fields [6], maximally stable regions [3], and pictorial structures [1]. Text, however, possesses some natural higher-order structural properties (linearity, availability of lexicons) that can disambiguate the recognition of even very complicated visual instances and that are not fully exploited by existing frameworks.

Most previous papers on natural text recognition are organized as bottom-up pipelines that perform character detection followed by character recognition. This delays the incorporation of language models (e.g. lexicon or n-grams) till the very last stage of the pipeline and often such priors are enforced as post-processing, e.g. in a spell-checker manner [7, 8]. Such an approach helps to recover from a moderate amount of errors, but the accumulation of errors towards higher stages of the pipeline in more difficult cases leads to incorrect results, as language priors come "too late" and are unable to "salvage" the recognition process.

Fig. 1. The top row: Examples of difficult images that are correctly recognized by our model without attribute reasoning. The second and the third rows: the words that are recognized by our model once the word attribute variables are added. The bottom row: images that are not recognized by our method.

In this paper, we present a text recognition (word reading) system that avoids the use of a pipeline architecture and instead formulates the problem of word recognition as the maximum a posteriori (MAP) inference in a unified probabilistic framework. Our model enforces both the language consistency, and the consistency of the attributes of letters that constitute a word. This unified treatment allows us to carry over the uncertainty associated with character detection and recognition in a principled fashion while enforcing structure and language constraints.

The MAP inference problem in our model requires potent optimization techniques, and for that we resort to *weighted finite-state transducers*, which are one of the most powerful frameworks that have been developed in the natural language/speech processing community. WFSTs provide a common and natural representation for recognition context, lexicons and grammars. Efficient algorithms for finding n-shortest paths, as well as for the *determinization* and the combination of WFSTs have been proposed in the literature [9, 10]. We utilize these as building blocks within the proposed inference procedure to ensure low computational cost even in the presence of very large (language-scale) lexicon priors.

We apply our approach on the problem of cropped word recognition, also known as *word spotting* (in the case of the small lexicon) and as *robust reading* (in the case of the large lexicon). We evaluate the method on ICDAR 2003[1], SVT[2], and the ICDAR 2011[3] image datasets/challenges, and in all cases observe a considerable improvement in the recognition accuracy over the previously reported state-of-the-art.

[1] http://algoval.essex.ac.uk/icdar/Datasets.html
[2] http://vision.ucsd.edu/ kai/grocr/
[3] http://robustreading.opendfki.de/wiki/SceneText

2 Related Work

As an exhaustive review of the literature concerned with text understanding in natural photographs is not feasible due to space limitations, we focus on the most related works.

Character detection. Several approaches have been proposed for character recognition that can to some extent handle the challenges introduced by natural photographs. [3, 4, 11] use maximally-stable extremal regions (MSER) as character candidates and achieve a high recall. We follow this line of work and use MSER as the primitives in our probabilistic model. Alternative approaches (which could also be used within our framework) include using connected components resulting from *stroke transforms* of edge maps [5], or using sliding HOG-templates [1].

Probabilistic models. A number of probabilistic methods have been proposed for character detection and identification in natural photos. Most of these models are similar in spirit to part-based deformable models/pictorial structures [12] that are popular in computer vision. These methods work by enforcing pairwise relations between adjacent characters and use message-passing/dynamic programming for MAP inference. Although they can easily incorporate the bi-gram prior, introducing more powerful language priors (e.g. lexicons) is highly non-trivial.

Some approaches [13, 8] incorporate lexicon priors in a rather exhaustive manner, with the resulting complexity scaling linearly with the size of the lexicons (although sparsification heuristics within message passing can retain some of the efficiency). Others (such as [14, 2]) incorporate lexicons by extending message-passing from linear chains to *tries* encoding language lexicons. While such approaches scale sublinearly with the size of the lexicon, *tries* for natural languages can still be very large. Passing messages along all edges of a trie, as required by these methods can still be a slow process (unless a restricted scenario such as word spotting [1, 2] with small lexicon is considered).

Weighted Finite-State Transducers (WFST). WFSTs represent a popular framework for working with natural languages. They provide a more compact representation to large lexicons compared to tries [15]. The use of WFST for text recognition in natural photographs has, however, been limited. Beaufort and Mancas-Thillou [7] have applied WFSTs to impose the language model as a postprocessing step after character segmentation and recognition. The closest work to ours is the recent paper of Yamazoe et al. [16]. Similarly to our work they combine two WFSTs: one to express the observed set of connected components and their properties, and the other to express the lexicon prior. Their approach, however, neither considers a joint energy/probabilistic model nor does it have latent variables (attributes).

As WFST forms the backbone of our approach we review it in more detail in the next sections. In the remainder of the paper, we establish the usefulness of latent variables, demonstrate the manner in which they can be handled efficiently with the determinization algorithms, and finally show that WFSTs can be used obtain excellent performance that exceeds state-of-the-art word recognition performance on established benchmarks.

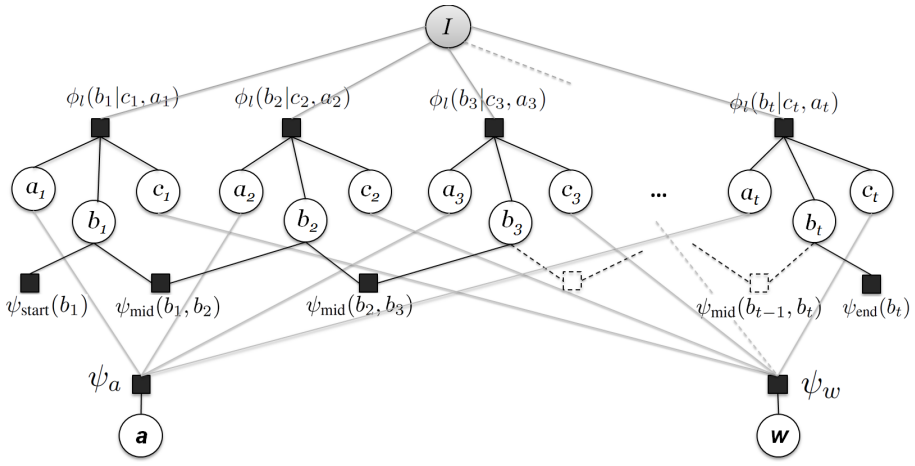


Fig. 2. The graphical model for word recognition. Here $\mathbf{w}$ is the result of the word recognition; $\mathbf{a}$ is the attribute-tuple of the word. The nodes $a_1, \ldots, a_t$ correspond to the attributes of the characters; $b_1, \ldots, b_t$ correspond to the character locations; $c_1, \ldots, c_t$ denote character recognition results (e.g. an alphanumeric symbol). The unary factors defined over a_i's and c_i's are omitted for clarity. The shaded node I denotes the observed image. Please, see text for more details.

3 Probabilistic model for word recognition.

Probabilistic formulation. Let Σ denote the alphabet set composed of symbols $\alpha_i; i \in \{1, ..., k\}$ (e.g. Latin letters and digits). Given a cropped image $I \in \mathcal{I}$ containing a certain word that fills most of this image, we formulate the goal of word recognition as the task of inferring this word $\mathbf{w}$, while possessing the dictionary (the *lexicon*) $\mathcal{L}$ of correct words. We now derive a model, so that the inference in such model accomplishes the goal in a robust manner.

In the beginning, to simplify the exposition, let us assume that the length t of the output word is known in advance and fixed. To determine which word is shown in the image, we then have to infer character locations $\{b_1, ..., b_t\}$ and character labels c_i at each of the $b_i \in \{1, ..., t\}$ locations. The summary of the visual information in the test image I at location b_i is used as a feature for character recognition. The set of all b_i's is denoted by B.

Each character c_i has some associated attributes a_i that characterize its appearance such as shape, color, size, stroke width, style, font etc. In our model, we restrict the attribute set to $a_i = \{a_i^c, a_i^o\}$ where a_i^c denotes the color of a character, a_i^o denotes its position with respect to the text line. We will use C and A to denote the set of all characters and attributes, i.e. $C = (c_1, ..., c_t)$ and $A = \{a_1, ..., a_t\}$.

Due to the typographic properties of the printed text, the values of attributes are usually the same or very similar for all characters constituting a word. For example, in most cases letters in a word have the same font, style and color. Moreover, the letters are usually organized in a text line, which provides a constraint on the locations and

sizes of the characters in an image. In order to capture such higher-order constraints we introduce a variable $\mathbf{a}$ that represents the attributes of the word $\mathbf{w}$ as a whole.

Given an image I, the Maximum a Posteriori (MAP) estimate of the word depicted in the image can be computed as:

$$\{\mathbf{w}^*, \mathbf{a}^*\} = \arg\max_{\mathbf{w},\mathbf{a}} P(\mathbf{w}, \mathbf{a}|I) = \arg\min_{\mathbf{w},\mathbf{a}} E(\mathbf{w}, \mathbf{a}, I) \tag{1}$$

where the energy E is the negative log-posterior for the distribution $P(\mathbf{w}, \mathbf{a}|I)$. The joint posterior distribution $P(\mathbf{w}, \mathbf{a}|I)$ of the word recognition result $\mathbf{w}$ and the word attribute $\mathbf{a}$ factorizes as:

$$\max_{\mathbf{w},\mathbf{a}} P(\mathbf{w}, \mathbf{a}|I) \propto \max_{\mathbf{w},\mathbf{a},A,B,C} \prod_i \phi_l(b_i|c_i, a_i)\psi_b(B)\psi_a(\mathbf{a}, A)\psi_w(\mathbf{w}, C). \tag{2}$$

where ϕ_l is the likelihood term, ψ_a is the attribute-consistency term, ψ_w corresponds to the language prior, ψ_b is the location prior. The labelling of all characters C and their attributes A are treated as latent variables.

In our model, the location prior ψ_b has the following form:

$$\psi_b(B) = \psi_{\text{start}}(b_1)\ \psi_{\text{end}}(b_t) \prod_{ij\in\mathcal{N}} \psi_{\text{mid}}(b_i, b_j) \tag{3}$$

where the term $\psi_{\text{mid}}(b_i, b_j)$ encourages the locations of adjacent characters $(i, j) \in \mathcal{N}$ to be consistent with each other, while the terms $\psi_{\text{start}}(b_1)$ and $\psi_{\text{end}}(b_t)$ encourage the locations of the first and the last characters to be close to the sides of the image (so that the entire image is approximately spanned by the word).

The attribute consistency term encourages the attributes of individual characters to be consistent with the attributes of the word $\mathbf{a}$ and, consequently, to be consistent with each other. It further decomposes into:

$$\psi_a(\mathbf{a}, A) = \prod_i \theta_c(\mathbf{a}, a_i) \prod_i \theta_o(\mathbf{a}, a_i) \tag{4}$$

where θ_c characterizes the color consistency of individual characters with the color of the word and θ_o characterizes the consistency of individual characters with the estimated text line.

The word prior $\psi_w(\mathbf{w}, C)$ encourages the characters detected at different locations to follow the language model. In our implementation it enforces the characters to form a word from the lexicon $\mathcal{L}$:

$$\psi_w(\mathbf{w}, C) = \begin{cases} 1, \text{ if } C \in \mathcal{L} \\ 0, \text{otherwise} \end{cases} \tag{5}$$

The corresponding graphical model (simplified for clarity) is shown in Figure 2.

Energy function. The energy (negative log likelihood) of our model if formally defined as:

$$\begin{aligned} E(\mathbf{w}, \mathbf{a}|I) = &\sum_i \tilde{\phi}_i^l(b_i|c_i, a_i) + \sum_i \tilde{\theta}_c(\mathbf{a}, a_i) + \sum_i \tilde{\theta}_o(\mathbf{a}, a_i) \\ &+ \tilde{\psi}_{\text{start}}(b_1) + \sum_{ij\in\mathcal{N}} \tilde{\psi}_{\text{mid}}(b_i, b_j) + \tilde{\psi}_{\text{end}}(b_t) + \tilde{\psi_w}(\mathbf{w}, C), \end{aligned} \tag{6}$$

where $\tilde{\phi}^l$, $\tilde{\theta}_c$, $\tilde{\theta}_o$ and $\tilde{\psi}$ are the negative logarithms of the potentials ϕ^l, θ_c, θ_o and ψ respectively.

Likelihood term. We compute the likelihood term ϕ_l using a simple nearest neighbour classifier with the shape descriptor similar to the one used in [3]. Let us denote the result of nearest neighbour classification for a character at location b_i by $\hat{c}_i$ and denote the distance from $I(b_i)$ to its nearest neighbour in the training set by $d(I(b_i), \hat{c}_i)$. Let $\nu(c', c'')$ denote the negative logarithm of the probability that a symbol c'' is classified as c'. We estimate $\nu(c', c'')$ by building a confusion matrix for nearest neighbour classification on the training set of synthetic characters.

The likelihood term $\tilde{\phi}_i^l$ in our model does not depend on the attributes of the character and takes the form:

$$\tilde{\phi}_l(b_i|c_i, a_i) = W_i \left[\kappa_{rec} d(I(b_i), \hat{c}_i) + \kappa_{conf} \left(1 - \nu(c_i, \hat{c}_i)\right)\right], \tag{7}$$

where κ_{rec} and κ_{conf} are parameters of the energy function, and W_i is the normalized width of the i-th character which is multiplied in the likelihood term to overcome bias towards shorter words. Note that there is a certain asymmetry between the nearest neighbour and other characters, which is introduced to allow for the efficient minimization discussed in section 4.

Prior terms. The prior terms for enforcing the attribute consistency have the following form:

$$\begin{aligned} \tilde{\theta}_c(\mathbf{a}, a_i) &= W_i \kappa_{\text{rgb}} \left[(r_i - r)^2 + (g_i - g)^2 + (b_i - b)^2\right] \\ \tilde{\theta}_o(\mathbf{a}, a_i) &= W_i \kappa_{\text{txl}} \times d^2_{\text{txl}} \left(x_i^t, y_i^t, x_i^b, y_i^b, \mathbf{a}_{\text{txl}}\right) \end{aligned} \tag{8}$$

where r_i, g_i, b_i is the mean color of the i-th character, $\{r, g, b\}$ is the mean color of the whole word, d_{txl} is the distance from the top and bottom points of the i-th character to the lines that bound the word (as computed in [4]), and κ_{rgb} and κ_{txl} are parameters of the energy function.

In our implementation the pairwise terms $\tilde{\psi}_{\text{mid}}(b_i, b_j)$ take a form:

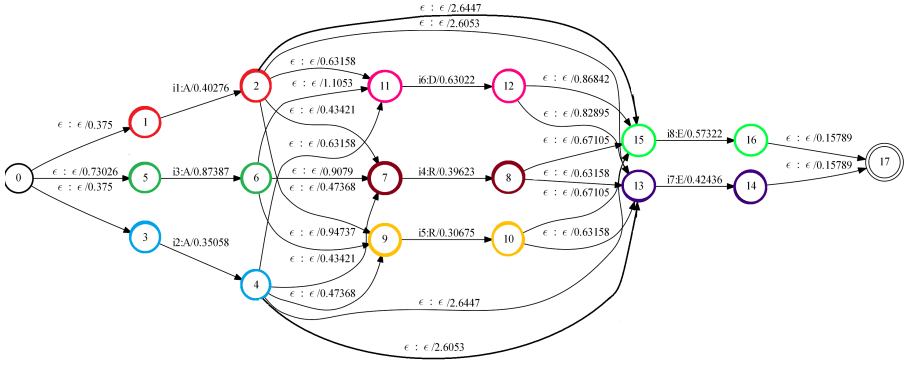
$$\tilde{\psi}_{\text{mid}}(b_i, b_j) = \begin{cases} W_{ij} \kappa_{\text{dist}}, \text{ if } W_{ij} < D_{max} \\ +\infty, \text{otherwise} \end{cases} \tag{9}$$

where W_{ij} is the normalized distance between the i-th and the j-th characters, thus ensuring that the consecutive characters are sufficiently close spatially. The potentials $\tilde{\psi}_{\text{start}}$ and $\tilde{\psi}_{\text{end}}$ have similar step-function form that guarantees that the positions of the first and last letters in the word are sufficiently close to the borders of the word bounding box. The constant κ_{dist} is a parameter of the energy function.

4 Inference using weighted finite state transducers

To perform inference in the model described in the previous section, we rely on classical language-related tools, namely weighted finite-state transducers (WFST) to express different parts of our model. We then use existing efficient implementations [17] to simplify, to *determinize*, and to compose these objects so that the parts of the model are

(a)- detected characters



(b)- initial transducer

Fig. 3. Constructing the initial transducer from sampled characters. (a) – a sample image with the sampled locations superimposed, (b) - the initial transducer. The nodes of the transducer are color-coded and correspond to detected characters. All characters that can be consecutive in the word are connected with transitions in the initial transducer. ϵ denotes the empty character.

mapped into a single WFST. We finally find the lowest-weight sequence of the resulting WFST that delivers the approximate MAP solution of (2).

In general, a weighted finite-state transducer (WFST) is based on a directed multi-graph, where each vertex corresponds to a state, and each arc is assigned a weight w, an input character c_i from the alphabet $\mathcal{A}_I$ and an output character c_o from the alphabet $\mathcal{A}_O$ (the notation $(c_i : c_o / w)$ is used further on). In this way, every path in the graph defines an *input sequence* of characters, an *output sequence* of characters, and a weight. One of the vertices of WFST is designated as a start vertex and a subset of vertices are designated as end vertices, thus defining a set of *valid* paths going from the start to one of the ends.

In this way, a WFST determines an input *language* $\mathcal{L}_I$ and an output language $\mathcal{L}_O$, defined as sets of character sequences corresponding to *valid* paths. Each valid path is said to *accept* its input sequence L_I and to *transduce* it to the output L_O. A WFST is called *deterministic*, if for each input-output sequence pairs there exists at most one valid path that performs the correspondent transduction. Each non-deterministic WFST A can be transformed to a deterministic WFST B with the same input and output languages *i. e.* for each transduction L_I into L_O that is possible in A, the output B has a single valid path with the same weight as the lowest-weight valid path performing this transduction in A. Such a transformation is called *determinization*, and a lot of research effort has gone into the development of efficient determinization algorithms that can keep the run-time and the size of the output WFST small.

Expressing the model via WFSTs. To perform the MAP inference in (2), we rely on the discretization process that samples likely values for the variables. Thus, we sample a large pool of the likely candidate character locations $\hat{b}_1, \hat{b}_2, \ldots, \hat{b}_N$ and the candidate word attribute values $\hat{\mathbf{a}}_1, \hat{\mathbf{a}}_2, \ldots, \hat{\mathbf{a}}_K$ (we discuss the sampling method below). For each candidate location $\hat{b}_i$, based on the local appearance, the most likely character $\hat{c}_i$ and the most likely character attribute value $\hat{a}_i$ are estimated.

Let us first consider a fixed word-level candidate attribute value $\hat{\mathbf{a}}$. We then create the WFST that transduces the sequence from the input alphabet $\{1, 2, \ldots, N\}$ into the output alpha-numerical alphabet. The graph for such a WFST is created in the following way (Figure 3). For each candidate character location, we create a *head* vertex H_i and a *tail* vertex T_i. We then connect the head and the tail vertices with the arc:

$$H_i \to T_i : \quad \left(i \,:\, \hat{c}_i \,/ W_i \kappa_{\text{rec}} d(I(\hat{b}_i), \hat{c}_i) + \tilde{\theta}_c(\hat{\mathbf{a}}, \hat{a}_i) + \tilde{\theta}_o(\hat{\mathbf{a}}, \hat{a}_i) \right), \tag{10}$$

where index i is the input character of the WFST, $\hat{c}_i$ is the output character, and the last parameter is the transduction weight. Note that one can introduce multiple arcs $H_i \to T_i$ corresponding to different characters that fit the appearance at the location $\hat{b}_i$ with some high likelihood. However, to reduce the computational complexity, we introduce only a single arc $H_i \to T_i$ corresponding to the most likely character at this stage, while the alternative characters are introduced at the later stage.

We then further detect all pairs of candidate locations $(\hat{b}_i, \hat{b}_j)$, which are sufficiently close to each other ($W_{ij} < D_{max}$). We then connect the tail T_i and the head H_j with the arc that accepts and produces empty characters (denoted with ϵ):

$$T_i \to H_j : \quad \left(\epsilon \,:\, \epsilon \,/ \tilde{\psi}_{\text{mid}}(\hat{b}_i, \hat{b}_j) \right). \tag{11}$$

We finally introduce the special start vertex S and the end vertex T. We introduce the set of arcs connecting S to the head vertices of the candidate locations situated near the left side of the bounding box with the parameters $\left(\epsilon \,:\, \epsilon \,/ \tilde{\psi}_{\text{start}}(\hat{b}_1) \right)$, and similarly connect the tail vertices of the locations near the right side to the T vertex with arcs having the parameters $\left(\epsilon \,:\, \epsilon \,/ \tilde{\psi}_{\text{end}}(\hat{b}_t) \right)$. The resulting WFST denoted as $\mathcal{T}'(\hat{\mathbf{a}})$ transforms the sequences of candidate locations into sequences of characters, with each such transformation being assigned a weight that is equal (upto an additive constant) to $\log \left(\prod_i \phi_l(\hat{b}_i | \hat{c}_i, \hat{a}_i) \psi_b(\mathbf{b}) \right)$.

"Determinizing out" the attribute variable. We handle the optimization over the attribute variable $\hat{\mathbf{a}}$ by creating a separate WFSTs $\mathcal{T}'(\hat{\mathbf{a}}_1), \mathcal{T}'(\hat{\mathbf{a}}_2), \ldots, \mathcal{T}'(\hat{\mathbf{a}}_K)$ for every candidate attribute value. We then combine all these WFSTs into a single WFST by merging their start and end vertices. An example of a transducer at this step is shown in Figure 4 (a). A further simplification/discretization step is applied to the WFST, a large number (2000 in our experiments) of valid paths with the lowest weight are found, and the part of the WFST that does not participate in those paths is discarded. The resulting WFST can still be non-deterministic (as the same transformation can be performed by multiple paths corresponding to different $\hat{\mathbf{a}}_i$). Therefore, the determinization algorithm is applied producing an equivalent deterministic WFST. Essentially, such determinization "minimizes out" the word-level attribute variable $\mathbf{a}$. We observe that the size of the

WFST is reduced dramatically in the determinization process. An example of WFST created after deteminization is shown in Figure 4 (b).

Reintroducing non-optimal characters. To handle the errors arising from adding only the most likely characters for each location $\hat{b}_i$ into the initial set of WFSTs, we modify the determinized WFST in the following way. Consider a head-tail transition (an arc) a with the parameters $(i : c_i / w_i)$. For each such arc, we add several parallel arcs corresponding to different possible substitutions c_i with parameters $(i : c_i / w_i^{c_i})$. The weight $w_i^{c_i}$ is set to $w_i + W_i \kappa_{\text{conf}} (1 - \nu(c_i, \hat{c}_i))$ so that the potential (7) is correctly implemented for all characters. The asymmetry of this potential w.r.t. the nearest neighbour and its independence from the attribute variable allows us to perform the determinization and to minimize out the attribute variable on a significantly smaller transducer. The transducer obtained after introducing the non-optimal characters is denoted as $\mathcal{T}_{\text{likelihood}}$. This transducer is usually very large, so we show just a small part of it in Figure 4 (b) in the balloon.

Adding the language prior. The main benefit of applying the WFST framework to the text recognition problem is the ease of introducing language priors. In this work, we focus on the more challenging lexicon priors, but the N-gram-based priors or the combination of multiple priors can be added as well. To add the lexicon prior $\psi_w(\mathbf{w}, C)$, we consider a standard lexicon finite-state transducer that accepts only the words from the lexicon corpus $\mathcal{L}$ [9] and apply the identity transformation to them (i.e. the WFST can be regarded as an *acceptor*). This WFST can also be unweighed, i.e. all transitions can be assigned a zero weight so that all words in the lexicon are assigned the same probability.

The transducer $\mathcal{T}_{\text{likelihood}}$ can then be composed [9] with the language-prior acceptor $\mathcal{T}_{\text{prior}}$. The result of the *composition* is a new transducer that transforms the sequences of candidate locations into words from the lexicon. To get the MAP-solution, one simply needs to find the shortest valid path in the composed transducer that accepts the optimal location sequence and produces the optimal word. The resulting transducer after the composition with the lexicon and the shortest path computation is shown in Figure 4 (c).

5 Experiments

Implementation details. As discussed earlier, our method needs to sample a large number of candidate locations and attributes for inference. For sampling these candidate assignments we follow the approach of [3] and use the MSER region detector to find location candidates (we use the implementation [18] with default parameters that provide sufficient recall). The appearance descriptor for the location region (MSER) is then computed using a 4x4 grid superimposed over its bounding box. Within each cell an 8-dimensional histogram of boundary orientations is computed and histograms for multiple cells are concatenated. Again, following [3] we create a large synthetic dataset of characters using Windows fonts resulting in 450 exemplars of each character. The dataset of the descriptors computed for such images is then used inside the nearest-neighbor classifier.

To sample attribute hypotheses, we sample 2 or 3 characters that are sufficiently close to each other and use this set for the estimation of mean color of the word and the textline parameters. The color of a word is estimated as a pixel-wise mean of the colors of all sampled characters. For the estimation of textline parameters we use the same strategy as suggested in [4]. If the slope of the textline is far from being horizontal ($> \mathrm{atan}0.4$), we discard the sequence and sample a different set of characters. This procedure is repeated until the total number of sampled word attributes is 100. For the inference in all experiments (and for the visualization in Figure 4 and Figure 3 we use the OpenFST library [17].

For all our experiments, the output character alphabet included the 26 Latin letters (case insensitive) as well as 10 digits. For the ICDAR'2011 experiments case-disambiguation is required, and we handled it in a post-hoc manner. Thus, we trained 26 linear SVMs, each one distinguishing uppercase English letter from the same lowercase letter. The SVMs are trained on a set of characters from Windows fonts. In most cases printed words either contain only lowercase/uppercase letters or have the first letter in the upper case followed by a tail of lowercase letters. To distinguish between these three classes of word layout, we computed the SVM margins for all letters in the word. We used the mean margin of all letters except for the first one to estimate the case of the tail. If the tail of the word is uppercase, we assumed that the first letter is also uppercase. Otherwise we estimate the case of the first letter based on the output of a corresponding SVM.

Parameter validation. In all reported experiments we used the same set of parameters, obtained by validation on the training part of ICDAR 2011 dataset. For validation we used the English lexicon from the OCRopus open source document analysis and OCR system [19] augmented with the list of words from the ICDAR 2011 train dataset (about 90 thousand words in total). We performed validation in two steps: first we performed validation of our model without word attributes and obtained the set of parameters κ_{rec}, κ_{conf}, κ_{dist}. After that we fixed these parameters and found optimal values of κ_{rgb} and κ_{txl}.

Word spotting scenario. In the first experiment we evaluate cropped word recognition on ICDAR 2003 and SVT datasets in the word spotting scenario (small lexicon) following the experimental protocol of [1] and [2]. For ICDAR 2003 the lexicon that contains all words that appear in the dataset (1065 words in total) is constructed. The SVT dataset provides a lexicon of 50 words for each image separately. Following the protocol used in [1] and [2] we ignore all words that contain non-alphanumeric characters as well as words with 2 or fewer characters.

In Table 1, we compare the results of our method to the methods from [1] and [2]. Our method performs 20% better on ICDAR 2003 and 14% better on SVT datasets than the closest competitor. Figure 1 shows examples of image from the ICDAR 2003 dataset.

In order to measure the contribution of different components of our system we perform additional experiments on ICDAR 2003, in which we evaluate two simplified variants of the method. In the first variant we remove the terms $\tilde{\theta}_c$ and $\tilde{\theta}_o$ that enforce attribute consistency. As seen in Table 1 including attribute consistency results in the significant increase in accuracy. In the second variant we consider the energy without

Table 1. Comparison of variants of our method with [1] and [2] in word spotting scenario on ICDAR'2003.

	ICDAR	SVT
Wang et al. [1]	59%	59%
Wang et al. [2]	62%	57%
Ours, full model	**82.8%**	**72.9%**
Ours w/o attribute consistency, w/o alternative character recognition	60%	-
Ours w/o attribute consistency	80.7%	-
Ours, full model, large lexicon	78.5%	-

the terms $\tilde{\theta}_c$ and $\tilde{\theta}_o$, and use only the most likely result of character recognition for each detected character. A substantial drop in performance suggests the importance of keeping multiple hypothesis for each character.

For comparison we measure the performance of our method with OCRopus lexicon augmented with the list of words from ICDAR 2003. The result of this experiment is also shown in Table 1. 90 times increase in the size of lexicon results in just about 4% decrease in accuracy. For comparison, in [2] increasing the size lexicon about 20 times (from 50 to 1065 words) led to a 14% performance decrease.

We have measured the running time of different components of our method. The method for sampling candidate characters and their attributes was implemented in MATLAB. The results for ICDAR 2003 dataset are summarized in Table 2. The suggested WFST-based inference is computationally efficient; in practice, the overall time is dominated by the nearest-neighbor classifier (not shown in the table) that can be optimized or replaced with e.g. Random Forest-based character recognition [20].

Table 2. Average running time of different components of our method (per word) for the different modules within the proposed inference procedure.

Sampling the attributes	0.11 sec
Determinization	0.27 sec
Adding alternatives for character recognition	0.26 sec
Composition with lexicon (about 90000 words) and shortest path computation	0.68 sec
Composition with lexicon (about 1000 words) and shortest path computation	0.21 sec

Robust reading scenario. The Robust Reading Competition was held at the International Conference on Document Analysis and Recognition in 2011. One of the challenges that it dealt with was the task of cropped word recognition. The target of this task was to recognize cropped word images of scene text. Cropping was done based on ground-truth word bounding boxes in order to evaluate recognition performance independently from text localization accuracy.

The lexicon used in the competition was not published but it is guaranteed to contain all words from the dataset. We used the English lexicon from the OCRopus open source document analysis and OCR system [19] that contains about 90 thousand words, which

is likely to be at least as large as the lexicon used in the challenge[4]. In order to place our method on the same footing as the other entries of the contest we added the words from the ICDAR 2011 dataset to this lexicon.

Table 3 shows the recognition rate for all methods. The results of the entries of the Robust Reading Competition are reproduced from [21]. Our method shows a very considerable 14% improvement over the winning entry of the Robust Reading Competition.

We also report the case-insensitive recognition rate. The gap between case-sensitive and case-insensitive recognition rates shows that the improvement of case recognition would result in further improvement of the recognition performance (although some instances in the dataset are inherently ambiguous or even erroneously annotated w.r.t. the letter case).

Table 3. Recognition rates of the entries of ICDAR 2011 Robust Reading Competition and our method for the ICDAR 2011 dataset.

	ICDAR 2011
TH-OCR System	41.2%
KAIST AIPR System	35.6%
Neumann's Method	33.11%
Our method (case sensitive)	**56.4**%
Our method (case insensitive)	*66.7%*

6 Discussion

We have proposed a novel method for text recognition that simultaneously models visual and lexicon consistency in a single probabilistic model. Unlike traditional stage-based methods, character and words recognition in our model is performed by estimating the MAP solution under a joint posterior distribution of the model. We have exploited the structure of our model to significantly speed-up MAP inference using weighted finite state transducers.

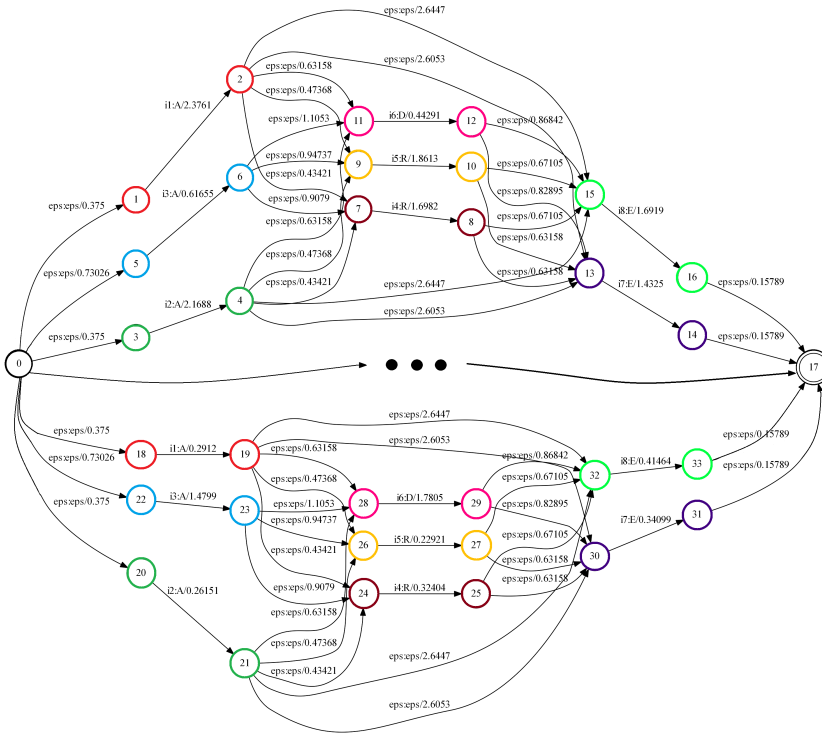
Experimental results demonstrate that our approach achieves substantially superior results compared to traditional methods. We have also shown that different parts of our model, such as attribute consistency, contribute significantly to the text detection performance.

References

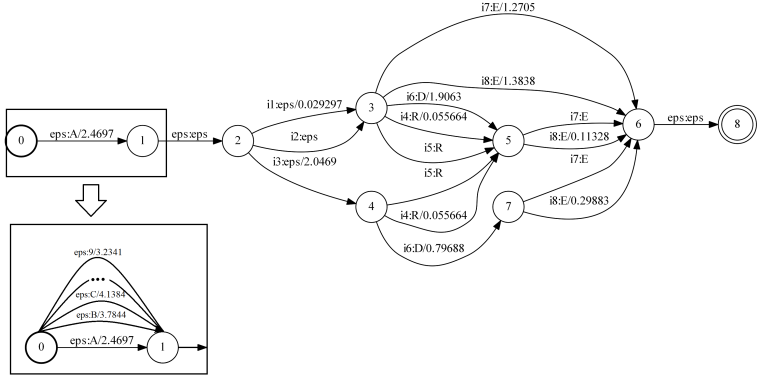
1. Wang, K., Belongie, S.: Word spotting in the wild. In: European Conference on Computer Vision (ECCV). (2010)
2. Wang, K., Babenko, B., Belongie, S.: End-to-end scene text recognition. In: IEEE International Conference on Computer Vision (ICCV). (2011)

[4] Note that apart from the language corpus, we have not used any component of the OCRopus system.

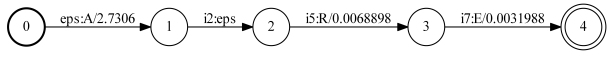
3. Neumann, L., Matas, J.: A method for text localization and recognition in real-world images. In: Proceedings of the 10th Asian conference on Computer vision - Volume Part III. ACCV'10 (2011)
4. Neumann, L., Matas, J.: Text localization in real-world images using efficiently pruned exhaustive search. In: ICDAR. (2011)
5. Epshtein, B., Ofek, E., Wexler, Y.: Detecting text in natural scenes with stroke width transform. In: CVPR. (2010)
6. Yuille, A.L.: Detecting and reading text in natural scenes. In: CVPR, IEEE (2004) 366–373
7. Beaufort, R., Mancas-Thillou, C.: A weighted finite-state framework for correcting errors in natural scene ocr. In: Proceedings of the Ninth International Conference on Document Analysis and Recognition - Volume 02. (2007)
8. Smith, D.L., Field, J., Learned-Miller, E.G.: Enforcing similarity constraints with integer programming for better scene text recognition. In: CVPR, IEEE (2011)
9. Mohri, M., Pereira, F., Riley, M.: Weighted finite-state transducers in speech recognition. Computer Speech & Language **16** (2002) 69–88
10. Povey, D., Hannemann, M., Boulianne, G., Burget, L., Ghoshal, A., Janda, M., Karafit, M., Kombrink, S., Motlcek, P., Qian, Y., Riedhammer, K., Vesel, K., Vu, N.T.: Generating exact lattices in the WFST framework. In: ICASSP. (2012)
11. Neumann, L., Matas, J.: Real-time scene text localization and recognition. In: CVPR. (2012)
12. Felzenszwalb, P.F., Huttenlocher, D.P.: Efficient belief propagation for early vision. In: CVPR (1). (2004) 261–268
13. Weinman, J.J., Learned-Miller, E., Hanson, A.R.: Scene text recognition using similarity and a lexicon with sparse belief propagation. IEEE Trans. Pattern Anal. Mach. Intell. **31** (2009)
14. Jacobs, C.E., Simard, P.Y., Viola, P.A., Rinker, J.: Text recognition of low-resolution document images. In: ICDAR. (2005) 695–699
15. Ciura, M., Deorowicz, S.: How to squeeze a lexicon. Softw., Pract. Exper. **31** (2001) 1077–1090
16. Yamazoe, T., Etoh, M., Yoshimura, T., Tsujino, K.: Hypothesis preservation approach to scene text recognition with weighted finite-state transducer. In: ICDAR. (2011)
17. Allauzen, C., Riley, M.: OpenFst: a general and efficient weighted finite-state transducer library. http://www.openfst.org/twiki/bin/view/FST/WebHome (2010)
18. Vedaldi, A., Fulkerson, B.: VLFeat: An open and portable library of computer vision algorithms. http://www.vlfeat.org/ (2008)
19. : The OCRopus open source document analysis and OCR system. (http://code.google.com/p/ocropus/)
20. Amit, Y., Geman, D.: Shape quantization and recognition with randomized trees. Neural Computation **9** (1997) 1545–1588
21. Shahab, A., Shafait, F., Dengel, A.: ICDAR 2011 robust reading competition challenge 2: Reading text in scene images. In: ICDAR. (2011)



(a)- transducer after sampling word attributes



(b)- transducer after determinization, reintroducing non-optimal characters is shown in a balloon



(c)- transducer after composition with lexicon and shortest path computation

Fig. 4. Components of the method for the example from Figure 3. Each branch in the trasnducer (a) corresponds to one sampled value of word attributes. After determinization for each input sequence only one valid path is kept, corresponding to the best value of the word attributes. After the composition with the lexicon only the valid paths that correspond to the words from a given lexicon are kept. The final answer is determied by the lowest-weight path (shown) in the combined transducer.