Top-Down and Bottom-up Cues for Scene Text Recognition

Anand Mishra[1] Karteek Alahari[2] C. V. Jawahar[1]

[1] CVIT, IIIT Hyderabad, India [2] INRIA - WILLOW / École Normale Supérieure, Paris, France

Abstract

Scene text recognition has gained significant attention from the computer vision community in recent years. Recognizing such text is a challenging problem, even more so than the recognition of scanned documents. In this work, we focus on the problem of recognizing text extracted from street images. We present a framework that exploits both bottom-up and top-down cues. The bottom-up cues are derived from individual character detections from the image. We build a Conditional Random Field model on these detections to jointly model the strength of the detections and the interactions between them. We impose top-down cues obtained from a lexicon-based prior, i.e. language statistics, on the model. The optimal word represented by the text image is obtained by minimizing the energy function corresponding to the random field model.

We show significant improvements in accuracies on two challenging public datasets, namely Street View Text (over 15%) and ICDAR 2003 (nearly 10%).

1. Introduction

The problem of understanding scenes semantically has been one of the challenging goals in computer vision for many decades. It has gained considerable attention over the past few years, in particular, in the context of street scenes [3, 20]. This problem has manifested itself in various forms, namely, object detection [10, 13], object recognition and segmentation [22, 25]. There have also been significant attempts at addressing all these tasks jointly [14, 16, 20]. Although these approaches interpret most of the scene successfully, regions containing text tend to be ignored. As an example, consider an image of a typical street scene taken from Google Street View in Figure 1. One of the first things we notice in this scene is the sign board and the text it contains. However, popular recognition methods ignore the text, and identify other objects such as car, person, tree, regions such as road, sky. The importance of text in images is also highlighted in the experimental study conducted by Judd *et al*. [17]. They found that viewers fixate on text when shown images containing text and other objects. This is further evidence that text recognition forms a useful component of the scene understanding problem.

Figure 1: *A typical street scene image taken from Google Street View [29]. It contains very prominent sign boards (with text) on the building and its windows. It also contains objects such as car, person, tree, and regions such as road, sky. Many scene understanding methods recognize these objects and regions in the image successfully, but tend to ignore the text on the sign board, which contains rich, useful information. Our goal is to fill-in this gap in understanding the scene.*

Given the rapid growth of camera-based applications readily available on mobile phones, understanding scene text is more important than ever. One could, for instance, foresee an application to answer questions such as, "*What does this sign say?*". This is related to the problem of Optical Character Recognition (OCR), which has a long history in the computer vision community. However, the success of OCR systems is largely restricted to text from scanned documents. Scene text exhibits a large variability in appearances, as shown in Figures 1 and 2, and can prove to be challenging even for the state-of-the-art OCR methods.

A few recent works have explored the problem of detecting and/or recognizing text in scenes [4, 6, 7, 11, 23,

Figure 2: *Scene text often contains examples that have a large variety of appearances. Here we show a few sample images from the* SVT *[30] and* ICDAR *[1] datasets, with issues such as, very different fonts, shadows, low resolution, occlusions. These images are much more complex than the ones seen in typical* OCR *datasets. Standard off-the-shelf* OCR*s perform very poorly on these datasets [23, 29].*

26, 29, 30, 31]. Chen and Yuille [6] and later, Epshtein *et al*. [11] have addressed the problem of detecting text in natural scenes. These two methods achieve significant detection results, but rely on an off-the-shelf OCR for subsequent recognition. Thus, they are not directly applicable for the challenging datasets we consider. De Campos *et al*. [9] proposed a method for recognizing cropped scene text characters. Although character recognition forms an essential component of text understanding, extending this framework to recognize words is not trivial. Weinman *et al*. [31] and Smith *et al*. [26] showed impressive scene text recognition results using similarity constraints and language statistics, but on a simpler dataset. It consists of "roughly fronto-parallel" pictures of signs [31], which are quite similar to those found in a traditional OCR setting. In contrast, we show results on a more challenging street view dataset [29], where the words vary in appearance significantly. Furthermore, we evaluate our approach on over 1000 words compared to 215 words in [26, 31].

The proposed approach is more closely related to those in [23, 29, 30], which address the problem of simultaneously localizing and recognizing words. On one hand, these methods localize text with a significant accuracy, but on the other hand, their recognition results leave a lot to be desired. Since words in the scene text dataset have been localized with a good accuracy, we focus on the problem of recognizing words, given their location. This is commonly referred to as the cropped word recognition problem. Note that the challenges of this task are evident from the best published accuracy of only 56% on the scene text dataset [29]. The probabilistic approach we propose in this paper achieves an accuracy of over 73% under identical experimental settings.

Our method is inspired by the many advancements made in the object detection and recognition problems [8, 10, 13, 25]. We present a framework that exploits both bottom-up and top-down cues. The bottom-up cues are derived from individual character detections from the image. Naturally, these windows contain true as well as false positive detections of characters. We build a Conditional Random Field (CRF) model [21] on these detections to determine not only the true positive detections, but also what word they represent jointly. We impose top-down cues obtained from a lexicon-based prior, *i.e*. language statistics, on the model. In addition to disambiguating between characters, this prior also helps us in recognizing words.

The remainder of the paper is organized as follows. In Section 2 we present our character detection method. Our framework to build the random field model with a top-down lexicon-based prior on these detections is described in Section 3. We provide results on two public datasets and compare our method to related work in Section 4. Implementation details are also given in this section. We then make concluding remarks in Section 5.

2. Character Detection

The first step in our approach is to detect potential locations of characters in a word image. We propose a sliding window based approach to achieve this.

2.1. Sliding Window Detection

Sliding window based detectors have been very successful for challenging tasks, such as face [28] and pedestrian [8] detection. Although character detection is similar to such problems, it has its unique challenges. Firstly, there is the issue of dealing with a large number of categories (62 in all). Secondly, there is a large amount of inter-character and intra-character confusion, as illustrated in Figure 3. When a window contains parts of two characters next to each other, it may have a very similar appearance to another character. In Figure 3(a), the window containing parts of the characters 'o' can be confused with 'x'. Furthermore, a part of one character can have the same appearance as that of another. In Figure 3(b), a part of the character 'B' can be confused with 'E'. We have adopted an additional pruning stage to overcome some of these issues.

We consider windows at multiple scales and spatial locations. The location of the i^{th} window, l_i, is given by its center and size. The set $\mathcal{K} = \{c_1, c_2, ..., c_k\}$, denotes the set of character classes in the dataset, *e.g*. $k = 62$ for English characters and digits. Let ϕ_i denote the features extracted from a window location l_i. Given the window l_i, we compute the likelihood, $p(c_i|\phi_i)$, of it taking a label c_i for all the

(a) (b)

Figure 3: *Typical challenges in multi-class character detection. (a) Inter-character confusion: A window containing parts of the two o's is falsely detected as x. (b) Intra-character confusion: A window containing a part of the character B is recognized as E.*

classes in $\mathcal{K}$. In our implementation, we used Histogram of Gradient (HOG) features [8] for ϕ_i, and the likelihoods $p(\cdot)$ were learnt using a multi-class Support Vector Machine (SVM) [5]. Details of the training procedure are provided in Section 4.2.

This basic sliding window detection approach produces many potential character windows, but not all of them are useful for recognizing words. We discard some of the weak detection windows using the following pruning method.

2.2. Pruning Windows

For every potential character window, we compute a score based on: (i) classifier confidence; and (ii) a measure of the aspect ratio of the character detected and the aspect ratio learnt for that character from training data. The intuition behind this score is that, a strong character window candidate should have a high classifier confidence score, and must fall within some range of sizes observed in the training data. For a window l_i with an aspect ratio a_i, let c_j denote the character with the best classifier confidence value given by S_{ij}. The mean aspect ratio (computed from training data) for the character c_j is denoted by μ_{a_j}. We define a goodness score (GS) for the window l_i as:

$$\text{GS}(l_i) = S_{ij} \exp\left(-\frac{(\mu_{a_j} - a_i)^2}{2\sigma_{a_j}^2}\right), \qquad (1)$$

where σ_{a_j} is the variance of the aspect ratio for character class c_j in the training data. Note that the aspect ratio statistics are character-specific. A low goodness score indicates a weak detection, and is removed from the set of candidate character windows.

We then apply Non-Maximum Suppression (NMS), similar to other sliding window detection methods [13], to address the issue of multiple overlapping detections for each instance of a character. We select detections which have a high confidence score, and do not overlap significantly with any of the other stronger detections. We perform NMS after the aspect ratio pruning because wide windows containing many characters may suppress overlapping single character windows, when they are weaker.

We perform both the pruning steps conservatively, and only discard the obvious false detections. We believe that this bottom-up approach alone cannot address all the issues related to detecting characters. Hence, we introduce lexicon-based top-down cues to discard the remaining false positives. We also use these cues to recognize the word as described in the following section.

3. Recognizing Words

The character detection step provides us with a large set of windows potentially containing characters within them. Our goal is to find the most likely word from this set of characters. We formulate this problem in an energy minimization framework, where the best energy solution represents the ground truth word we aim to find.

3.1. The Word Model

Each detection window is represented by a random variable X_i, which takes a label x_i.[1] Let n be the total number of detection windows. Note that the set of random variables includes windows representing not only true positive detections, but also many false positive detections, which must be suppressed. We introduce a null (or void) label ϵ to account for these false windows. Thus, $x_i \in \mathcal{K}_\epsilon = \mathcal{K} \cup \{\epsilon\}$. The set $\mathcal{K}_\epsilon^n$ represents the set of all possible assignments of labels to the random variables. An energy function $E : \mathcal{K}_\epsilon^n \rightarrow \mathbb{R}$, maps any labelling to a real number $E(\cdot)$ called its energy or cost. The function $E(\cdot)$ is commonly represented as a sum of unary and pairwise terms as:

$$E(\mathbf{x}) = \sum_{i=1}^{n} E_i(x_i) + \sum_{\mathcal{E}} E_{ij}(x_i, x_j), \qquad (2)$$

where $\mathbf{x} = \{x_i | i = 1, 2, \ldots, n\}$, $E_i(\cdot)$ represents the unary term, $E_{ij}(\cdot, \cdot)$ is the pairwise term, and $\mathcal{E}$ represents the set of pairs of interacting detection windows, which is determined by the structure of the underlying graphical model.

Graph construction. We order the windows based on their horizontal location, and add one node each for every window sequentially from left to right. The nodes are then connected by edges. One could make a complete graph by connecting each node to every other node. However, it is not natural for a window on the extreme left to be related to another window on the extreme right, as evident in Figure 4.[2] Thus, we only connect windows with a significant overlap between them or windows which are close to each

[1] Our model has similarities to that proposed in [10] for object detection, but the challenges (*e.g.* inter- and intra- character confusions) are greatly different from those in the object detection problem.

[2] We assume here that the windows have been pruned based on aspect ratio of character windows. Without this pruning step, a window may contain multiple characters and will perhaps require a more complete graph.

other. In the example in Figure 4, we show a few window samples and the edges between them. The intuition behind connecting overlapping or close-proximity windows is that they could represent either overlapping detections of the same character or detections of two separate characters. As we will see later, the edges are used to encode the language model as top-down cues.

CRF energy. The unary term $E(x_i)$, which denotes the cost of a node x_i taking label $c_j \neq \epsilon$, is defined as:

$$E_i(x_i = c_j) = 1 - p(c_j|x_i), \tag{3}$$

where $p(c_j|x_i)$ is the classifier confidence (*e.g.* SVM score) of character class c_j for node x_i. For the null label ϵ,

$$E_i(x_i = \epsilon) = \max_j p(c_j|x_i) \exp\left(-\frac{(\mu_{a_j} - a_i)^2}{\sigma_{a_j}^2}\right), \tag{4}$$

where a_i is the aspect ratio of the window corresponding to node x_i, c_j is character detected at node x_i, and μ_{a_j} and σ_{a_j} are the mean and the variance of aspect ratio of the character detected, which is learnt from the training data, respectively. For a true window, which has a relatively good SVM score, and matches the average aspect ratio size, this cost of assigning a null label is high. On the other hand, false windows, which either have poor SVM scores or vary from the average aspect ratio size or both will be more likely to take the null label ϵ.

The pairwise term $E(x_i, x_j)$ is used to encode the top-down cues from the language model in a principled way. The cost of two neighbouring nodes x_i and x_j taking labels $c_i \neq \epsilon$ and $c_j \neq \epsilon$ respectively is given by:

$$E_{ij}(x_i, x_j) = E^l_{ij}(x_i, x_j) + \lambda_o \exp(-\psi(x_i, x_j)). \tag{5}$$

Here, $\psi(x_i, x_j) = (100 - \text{Overlap}(x_i, x_j))^2$, is a measure of the overlap percentage between the two windows X_i and X_j. The function $E^l_{ij}(x_i, x_j)$ denotes the lexicon prior.The parameter λ_o determines the overlap-based penalty. Computation of the lexicon prior $E^l_{ij}(\cdot, \cdot)$ is discussed in Section 3.2. The pairwise cost (5) ensures that two windows with sufficiently high overlap cannot take non-null labels, *i.e.* at least one of them is likely to be a false window. The costs involving the null label ϵ are computed as:

$$E_{ij}(x_i = c_i, x_j = \epsilon) = \lambda_o \exp(-\psi(x_i, x_j)). \tag{6}$$

The pairwise cost $E_{ij}(x_i = \epsilon, x_j = c_j)$ is defined similarly. Further, $E_{ij}(x_i = \epsilon, x_j = \epsilon)$ is uniformly set to zero.

Inference. Given these unary and pairwise terms, we minimize the energy function (2). We use the sequential tree-reweighted message passing (TRW-S) algorithm [18] because of its efficiency and accuracy on our recognition problem. The TRW-S algorithm maximizes a concave lower bound on the energy. It begins by considering a set of trees from the random field, and computes probability distributions over each tree. These distributions are then used to reweight the messages being passed during loopy BP [24] on each tree. The algorithm terminates when the lower bound cannot be increased further, or the maximum number of iterations has reached.

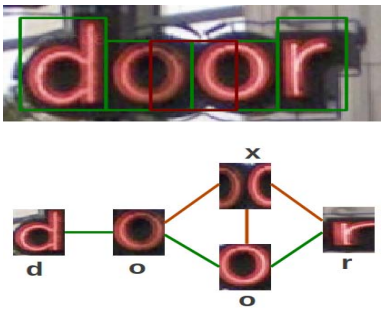

Figure 4: *Summary of our approach. We first find a set of potential character windows, shown at the top (only a few are shown here for readability). We then build a random field model on these detection windows by connecting them with edges. The edge weights are computed based on the characteristics of the two windows. Edges shown in green indicate that the two windows it connects have a high probability of occurring together. Edges shown in red connect two windows that are unlikely to be characters following one another. A edge shown in red forces one of the two windows to take the ϵ label, i.e. removes it from the candidate character set. Based on these edges and the* SVM *scores for each window, we infer the character classes of each window as well the word, which is indicated by the green edge path. (Best viewed in colour.)*

3.2. Computing the Lexicon Prior

We use a lexicon to compute the prior $E^l_{ij}(x_i, x_j)$ in (5). Such language models are frequently used in speech recognition, machine translation, and to some extent in OCR systems [27]. We explore two types of lexicon priors for the word recognition problem.

Bi-gram. Bi-gram based lexicon priors are learnt from joint occurrences of characters in the lexicon. Character pairs which never occur are highly penalized. Let $P(c_i, c_j)$ denote the probability of occurrence of a character pair (c_i, c_j) in the lexicon. The pairwise cost is:

$$E^l_{ij}(x_i = c_i, x_j = c_j) = \lambda_l(1 - P(c_i, c_j)), \tag{7}$$

where the parameter λ_l determines the penalty for a character pair occurring in the lexicon.

Node-specific prior. When the lexicon increases in size, the bi-gram model loses its effectiveness. It also fails to capture the location-specific information of pairs of characters. As a toy example, consider a lexicon with only two words CVPR and ICPR. The node-specific pairwise cost for the character pair (P,R) to occur at the beginning of the word is higher than for it to occur at the end of word. This useful cue is ignored in the bi-gram prior model.

To overcome this, we divide each lexicon word into n parts, where n is determined based on the number of nodes in the graph and the spatial distance between nodes. We then use only the first $1/n^{\text{th}}$ of the word for computing the pairwise cost between initial nodes, similarly next $1/n^{\text{th}}$ for computing the cost between the next few nodes, and so on. In other words, we do a region of interest (ROI) based search in the lexicon. The ROI is determined based on the spatial position of a detected window in the word, *e.g.* if two windows are on the left most side then only the first couple of characters of lexicons are considered for calculating the pairwise term between windows.

The pairwise cost using this prior is given by:

$$E^l_{ij}(x_i = c_i, x_j = c_j) = \begin{cases} 0 & \text{if } (c_i, c_j) \in \text{ROI}, \\ \lambda_l & \text{otherwise.} \end{cases} \quad (8)$$

We evaluate our approach with both these pairwise terms, and find that the node-specific prior achieves better performance.

4. Experiments

In this section we present a detailed evaluation of our method. Given a word image extracted from a street scene and a lexicon, our problem is to find all the characters, and also to recognize the word as a whole. We evaluate various components of the proposed approach to justify our choices. We compare our results with two of the best performing methods [29, 30] for this task.

4.1. Datasets

We used the Street View Text (SVT) [30][3] and the ICDAR 2003 robust word recognition [1] datasets in our experiments. To maintain identical experimental settings to those in [29], we use the lexica provided by them for these two datasets.

SVT. The Street View Text (SVT)[4] dataset contains images taken from Google Street View. As noted in [30], most of the images come from business signage and exhibit a high degree of variability in appearance and resolution. The dataset is divided into SVT-SPOT and SVT-WORD, meant for the tasks of locating words and recognizing words respectively. Since, in our work, we focus on the word recognition task, we used the SVT-WORD dataset, which contains 647 word images.

Our basic unit of recognition is a character, which needs to be detected or localized before classification. A miss in the localization will result in poorer word recognition. To improve the robustness of the recognition architecture, we need to quantitatively measure the accuracy of the character detection module. For this purpose, we created ground truth data for characters in the SVT-WORD dataset. Using the ground truth at the character level we evaluated the performance of the SVM classifier used for this task. Note that no such evaluation has been reported on the SVT dataset as yet. Our ground truth data set contains around 4000 characters of 52 classes overall. We refer to this dataset as SVT-CHAR.[5]

ICDAR 2003 Dataset. The ICDAR 2003 dataset was originally created for cropped character classification, full image text detection, cropped and full image word recognition, and various other tasks in document analysis. We used the part corresponding to cropped image word recognition called Robust Word Recognition [1]. Similar to [29], we ignore words with less than two characters or with non-alphanumeric characters, which results in 829 words overall. For subsequent discussion we refer to this dataset as ICDAR(50).

4.2. Character Detection

Sliding window based character detection is an important component of our framework, as our random field model is defined on the detections obtained. At every possible location of the sliding window, we test a character classifier. This provides a likelihood of the window containing a certain character. The alphabet of characters recognized consists of 26 lowercase and 26 uppercase letters, and 10 digits.

We evaluated various features for recognizing characters. We observed that the HOG feature [8] outperforms the features reported in [9], which uses a bag-of-words model. This is perhaps due to the lack of geometric information in the model. We computed dense HOG features with a cell size of 4×4 using 10 bins, after resizing each image to a 22×20 window. We learnt a 1-vs-all SVM classifier with an RBF kernel using these features. We used the standard LIBSVM package [5] for training and testing the SVMs. For the SVT-CHAR evaluation, we trained the model on the ICDAR 2003 dataset due to the relatively small size of SVT-CHAR ($\sim$ 4000 characters). We observed that the method using HOG descriptors performs significantly better than others with a classification accuracy of 61.86%.

[3] http://vision.ucsd.edu/$\sim$kai/svt

[4] Note that this dataset has been slightly updated since its original release in [30]. We use the updated version [29] in our experiments.

[5] Available at http://cvit.iiit.ac.in/projects/SceneTextUnderstanding

Total characters in the ground truth	3796
True detection (true positive)	3707
Characters we missed (false negative)	89

Table 1: *Evaluation of the performance of our sliding window approach. We use the intersection over union measure [1, 12] thresholded at 90% to determine whether a detection has been retrieved or not. Note that most of the true positives are detected. A few of them, such as the ones shown in Figure 5 were missed.*

We then performed sliding window based detection, where we considered bounding boxes of varying sizes, at many locations in the cropped word image. Each bounding box is classified with a certain likelihood given by its SVM score. We pruned some of the windows based on their aspect ratio. We used the goodness score measure in (1), and discarded all windows with a score less than 0.1. Character specific NMS is done on the remaining bounding boxes. This two step pruning is done to discard some of the obvious errors, although making sure that almost all the true positives are included at this stage. Table 1 summarizes an evaluation of the quality of our sliding window approach for the SVT-CHAR dataset. We computed the intersection over union measure of a detected window compared to the ground truth, similar to PASCAL VOC [12] and ICDAR 2003 [1]. We used a threshold of 90% to determine if a box has missed a character or not. Note that more than 97% of the characters are detected. Some of the typical failures are due to poor resolution of the images, which leads to very weak SVM scores, as shown in Figure 5. In addition to the detection rate, we also measured the performance of this module in terms of precision-recall. We obtained an mAP of 0.88 over all the characters, which indicates that this approach provides reliable cues for other parts of our framework.

4.3. Cropped Word Recognition

We use the detections obtained to build the CRF model as discussed in Section 3.1. We add one node for every detection window, and connect it to other windows based on its spatial distance and overlap. Two nodes spatially distant from each other are not connected directly. The unary cost of taking a character label is determined by the SVM confidence, and the pairwise costs are learnt from the lexicon. We experiment with both bi-gram and node-specific lexicon priors.

We choose the lexicon prior parameter in (7) and (8) $\lambda_l = 2$, for all our experiments. The overlap penalty parameter in (5) and (6) is set to $\lambda_o = 1$, empirically for all our experiments. The resulting energy function is optimized using the TRW-S algorithm [18].

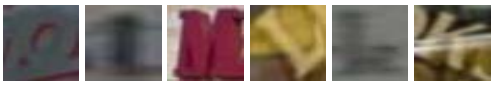

Figure 5: *A few challenging character examples we missed in the sliding window stage. These examples are very difficult even for a human. We observed that all these potential character windows were missed due to poor* SVM *scores.*

4.4. Results and Discussion

Similar to the evaluation scheme in [29], we use the inferred result to retrieve the word with the smallest edit distance in the lexicon. We evaluated our methods on two of the most challenging street scene text datasets, namely SVT and ICDAR 2003. Other databases, such as the ones used in [26, 31], mainly consist of "roughly fronto-parallel" pictures of signs [31], or do not consider low resolution or low contrast images [26]. This makes the images in these datasets quite similar to those found in a traditional OCR setting. Since our focus is to work with typical images from a street scene, we used the most challenging datasets available publicly.

The best known results on the SVT and ICDAR datasets are reported in [29], where the authors used a pictorial structures model for recognizing words. Under similar experimental settings, we compare our results with this and other methods in Table 2. We observe that our method outperforms [29] and the commercial OCR ABBYY [2].

We also study the effect of the two pairwise terms, namely bi-gram and node-specific priors. We note that the node-specific prior is superior to the bi-gram prior. As the number of words in the lexicon increases, bi-gram probability does not convey much information. Moreover, it fails to capture cues such as, some character pairs occur more frequently at the beginning of the word and very rarely towards the end.

We show the qualitative performance of our method in Figure 7. These examples demonstrate the benefits of combining top-down (node-specific lexicon) and bottom-up (character detection) cues in word recognition. It does have failures however. Some of the failure cases are shown in the last row in the figure. In our evaluation, we found the main causes of such failures to be: (i) weak unary term; and (ii) missing characters in the detection stage itself (Figure 5).

The increase in accuracy of more than 15% on the SVT-WORD dataset and nearly 10% on the ICDAR 2003 dataset can be attributed to the strengths of our model: (i) We perform better sliding window detection and get most of the true positives. Note that we quantitatively measure our performance of this stage and miss only 2% of the true windows. (ii) We use the context (*i.e.* lexicon prior) in a more principled way. (iii) We seamlessly integrate the bottom-up and top-down clues in a CRF framework.

Figure 6: *Bi-gram vs node-specific prior. The word inferred using bi-gram prior is shown on the left (in the blue box) and that inferred using node-specific prior is shown on the right (in the yellow box) at the top of every image. The node-specific prior shows a better performance over the bi-gram prior as it captures relative occurrence of characters more effectively. (Best viewed in colour.)*

Discussion. Our method is inspired by the work of [29] and [10], but we differ from them in many aspects as detailed below. The method of [29] is a strongly top-down approach, where each word in the lexicon is matched to a given set of character windows, and the one with the lowest score is the predicted word. This is prone to errors when characters are missed or detected with low scores. In contrast, we build a model from the entire lexicon (top-down cues), combine it with all the character detections (bottom-up cues), which have low or high scores, and infer the word with a joint inference scheme. This model, which uses both top-down and bottom-up cues, provides us with a significant improvement in accuracy. Furthermore, unlike the trie based structure for storing lexicon in [29], our method is not restricted by the size of the lexicon.

The structured label computed in [10] does not have any semantic meaning, and simply states that image contains certain objects. In our work, the inferred label represents a word from the lexicon, which requires use of much more complex context (top-down cues). It is not clear if [10] can incorporate such constraints easily.

5. Conclusion

In summary, we proposed an effective method to recognize scene text. Our model combines bottom-up cues from character detections and top-down cues from lexica. We infer the location of true characters and the word they represent as a whole jointly. We showed results on two of the most challenging scene text databases, and improved the latest results published at ICCV 2011 [29] significantly. Our results show that scene text can be recognized with a reasonably high accuracy in natural, unconstrained images. This could help in building vision systems, which can solve higher level semantic tasks, such as [15, 19].

Method	SVT-WORD	ICDAR(50)
PICT [30]	59.0*	-
PLEX + ICDAR [29]	56	72
ABBYY [2]	35	56
Proposed (Bi-gram)	70.03	76.96
Proposed (Node-specific)	**73.26**	**81.78**

Table 2: *Cropped Word Recognition Accuracy (in %): We show a comparison of the proposed random field model to the popular commercial* OCR *system* ABBY*, and* PLEX *proposed in [29]. *Note that the dataset has been revised since the original publication in [30], which makes that result not comparable directly. However, given that our method performs better than the improved version of [30], i.e. [29], we expect this trend to hold. We improve the accuracy by over 15% and 10% respectively on* SVT-WORD *and* ICDAR *datasets.*

Acknowledgements

This work is partly supported by the Ministry of Communications and Information Technology, Government of India, New Delhi. Karteek Alahari is partly supported by the Quaero programme funded by the OSEO and the EIT ICT Labs (activity 10863). We thank Armand Joulin for helpful comments on the manuscript.

References

[1] ICDAR 2003 Robust word recognition dataset, http://algoval.essex.ac.uk/icdar/RobustWord.html.

[2] ABBYY Finereader 9.0, http://www.abbyy.com.

[3] G. J. Brostow, J. Shotton, J. Fauqueur, and R. Cipolla. Segmentation and recognition using structure from motion point clouds. In *ECCV*, volume 1, pages 44–57, 2008.

[4] C. Case, B. Suresh, A. Coates, and A. Y. Ng. Autonomous sign reading for semantic mapping. In *ICRA*, 2011.

[5] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Trans. Intelligent Systems and Technology*, 2011.

[6] X. Chen and A. L. Yuille. Detecting and reading text in natural scenes. In *CVPR (2)*, pages 366–373, 2004.

[7] A. Coates, B. Carpenter, C. Case, S. Satheesh, B. Suresh, T. Wang, D. J. Wu, and A. Y. Ng. Text detection and character recognition in scene images with unsupervised feature learning. In *ICDAR*, pages 440–445, 2011.

[8] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.

[9] T. E. de Campos, B. R. Babu, and M. Varma. Character recognition in natural images. In *VISAPP*, 2009.

[10] C. Desai, D. Ramanan, and C. Fowlkes. Discriminative models for multi-class object layout. In *ICCV*, 2009.

[11] B. Epshtein, E. Ofek, and Y. Wexler. Detecting text in natural scenes with stroke width transform. In *CVPR*, 2010.

Figure 7: *Example word recognition results on challenging images from the* SVT *dataset [29] using the node-specific lexicon prior. Our method handles many different fonts, distortions (e.g.* CAPOGIRO *in row 3). There are few failure cases however, as shown in the bottom row. We found the main causes of such failures to be: (i) weak unary term; and (ii) missing characters in the detection stage itself (see Figure 5). Some of these issues (e.g.* CITY *image in the bottom row) can be addressed by modifying the sliding window procedure.*

[12] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes (VOC) challenge. *IJCV*, 88(2):303–338, 2010.

[13] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *PAMI*, 32(9), 2010.

[14] S. Gould, T. Gao, and D. Koller. Region-based segmentation and object detection. In *NIPS*, 2009.

[15] A. Gupta, Y. Verma, and C. V. Jawahar. Choosing linguistics over vision to describe images. In *AAAI*, 2012.

[16] D. Hoiem, A. Efros, and M. Hebert. Closing the loop on scene interpretation. In *CVPR*, 2008.

[17] T. Judd, K. Ehinger, F. Durand, and A. Torralba. Learning to predict where humans look. In *ICCV*, 2009.

[18] V. Kolmogorov. Convergent tree-reweighted message passing for energy minimization. *PAMI*, 2006.

[19] G. Kulkarni, V. Premraj, S. Dhar, S. Li, Y. Choi, A. C. Berg, and T. L. Berg. Baby Talk: Understanding and generating simple image descriptions. In *CVPR*, 2011.

[20] L. Ladicky, P. Sturgess, K. Alahari, C. Russell, and P. H. S. Torr. What, Where & How Many? Combining object detectors and CRFs. In *ECCV*, 2010.

[21] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labelling sequence data. In *ICML*, pages 282–289, 2001.

[22] A. Levin and Y. Weiss. Learning to combine bottom-up and top-down segmentation. *IJCV*, 81(1):105–118, 2009.

[23] L. Neumann and J. Matas. A method for text localization and recognition in real-world images. In *ACCV*, 2010.

[24] J. Pearl. *Probabilistic Reasoning in Intelligent Systems : Networks of Plausible Inference*. Morgan Kauffman, 1988.

[25] J. Shotton, J. Winn, C. Rother, and A. Criminisi. Textonboost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context. *IJCV*, 81(1):2–23, 2009.

[26] D. L. Smith, J. Field, and E. G. Learned-Miller. Enforcing similarity constraints with integer programming for better scene text recognition. In *CVPR*, 2011.

[27] R. Smith. Limits on the application of frequency-based language models to OCR. In *ICDAR*, 2011.

[28] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. *CVPR*, 2001.

[29] K. Wang, B. Babenko, and S. Belongie. End-to-end scene text recognition. In *ICCV*, 2011.

[30] K. Wang and S. Belongie. Word spotting in the wild. In *ECCV*, pages 591–604, 2010.

[31] J. J. Weinman, E. G. Learned-Miller, and A. R. Hanson. Scene text recognition using similarity and a lexicon with sparse belief propagation. *PAMI*, 2009.