A Text Detection System for Natural Scenes with Convolutional Feature Learning and Cascaded Classification

Siyu Zhu
Center for Imaging Science
Rochester Institute of Technology, NY, USA
zhu_siyu@hotmail.com

Richard Zanibbi
Department of Computer Science
Rochester Institute of Technology, NY, USA
rlaz@cs.rit.edu

Abstract

We propose a system that finds text in natural scenes using a variety of cues. Our novel data-driven method incorporates coarse-to-fine detection of character pixels using convolutional features (Text-Conv), followed by extracting connected components (CCs) from characters using edge and color features, and finally performing a graph-based segmentation of CCs into words (Word-Graph). For Text-Conv, the initial detection is based on convolutional feature maps similar to those used in Convolutional Neural Networks (CNNs), but learned using Convolutional k-means. Convolution masks defined by local and neighboring patch features are used to improve detection accuracy. The Word-Graph algorithm uses contextual information to both improve word segmentation and prune false character/word detections. Different definitions for foreground (text) regions are used to train the detection stages, some based on bounding box intersection, and others on bounding box and pixel intersection. Our system obtains pixel, character, and word detection f-measures of 93.14%, 90.26%, and 86.77% respectively for the ICDAR 2015 Robust Reading Focused Scene Text dataset, out-performing state-of-the-art systems. This approach may work for other detection targets with homogenous color in natural scenes.

1. Introduction

In natural scenes, text detection is made difficult by high variation in character color, font, size, and orientation. In addition, light sources introduce highlight, shadow, reflection and color offset in images, while cameras introduce additional noise, blurring, and viewing angle distortion. In the past, many text detection systems addressed this using strong prior knowledge and carefully engineered features [24]. More recently, machine learning methods are preferred over heuristic rules, with parameters and thresholds inferred automatically from training data. This requires less human intervention, and generally increases the accuracy and robustness of text detection.

As described by Ye et al. [24], step-wise text detection is composed of different components, including localization, verification, segmentation and sometimes recognition. 'Holistic' methods combine results from different stages, often applying OCR results for use in lexicon matching.

In this paper, we present a highly accurate text detection system for natural scenes utilizing only visual features.[1] Our system is composed of the *Text-Conv* algorithm for character patch detection, region growing to obtain Connected Components (CCs) corresponding to characters, and then word segmentation using the *Word-Graph* algorithm.

Contributions. The contributions of this work include: (1) defining ground truth character patches differently for coarse vs. fine character detection, first using constraints on bounding box intersection, and then bounding box and foreground pixel intersection; (2) using 'contextual' detection windows to improve discrimination based on adjacent and/or missing characters; (3) multi-stage generation and validation of character detections using convolutional, geometric and contextual features. Our system also obtains state-of-the-art performance for the challenging 2015 ICDAR Robust Reading Focused Scene Text dataset [12].

In Section 2, we briefly review state-of-the-art text detection systems, and the ICDAR Robust Reading Competition. In Section 3, we describe our system. In Section 4 we present experimental results, and then conclude and identify ways to accelerate and improve our system in Section 5.

2. Previous Work

In recent years, new discriminative features have been proposed for text detection, including the Stroke Width Transform (SWT) [7] and Maximal Stable Extremal Regions (MSER) [14], both of which have been used widely. Most characters have a narrow and uniform stroke width,

[1] Source code:
https://www.cs.rit.edu/~dprl/Software.html.

along with clear edges and homogeneous colors. SWT and MSER are designed to capture these properties.

A variety of machine learning techniques have been used for text detection, including unsupervised feature learning, Convolutional Neural Networks [13], deformable part-based models [8], belief propagation [9], and Conditional Random Fields [19]. Bai et al. [1] identify text regions using gradient local-correlation to find edge pairs and estimate stroke width. The relationship between different CCs, colors and shapes are fed into SVM classifiers to detect text.

Closely related to our own work, Coates et al. [5] proposed convolutional feature-based algorithms for text detection and recognition, and increased recognition rates relative to previous state-of-art systems. Wang et al. [22] extend this convolution-based approach with a lexicon model, which further increases text detection and recognition accuracy. Our text detection algorithm is based on this work; to increase accuracy, we modify the sliding window detector by varying the sliding window shape, rotation and aspect ratio.

2.1. ICDAR Robust Reading Competitions

Over the last two decades, a number of text recognition competitions have been held as part of the International Conference of Document Analysis and Recognition (ICDAR) [12]. In 2015, the task data sets are categorized into: Digital Born, Focused Scene, Text in Videos and Incidental Scene Texts. Each group contains three tasks: localization, segmentation and recognition, and end-to-end performance is also measured. We focus here on the scene text data set.

At ICDAR 2015, the StradVision corporation obtained the strongest detection results for the Focused Scene Text task. This system is closed, but from the company's web page appears to be based on active ('agile') learning.[2]

He et al. [10] placed second, using two main improvements over earlier MSER-based text detection methods. First, they introduce Text-CNN, where a multi-class classifier is defined instead of a conventional binary (text/non-text) classifier. In each layer of a Convolutional Neural Network, specific labels and locations for text pixels define targets alongside binary foreground/background labels. The trained classifier is adapted to specific text types, and achieves higher accuracy as a result. Objects like bricks, windows and bars that can be easily confused with text may be filtered, as they tend not to have a high classification confidence for a character class. Second, Contrast-Enhanced MSER is proposed to find text regions, using a data-driven contrast enhancement method before MSER, allowing text regions to be extracted in complex backgrounds and uneven lighting conditions.

Jaderberg et al. [11] placed third. To train their system, a large corpus of labeled text images is generated using a font rendering engine. Noise and variations are added, including border, color, composition, distortion, and background blending, mimicking texts in natural scenes. For detection, they use a deep Neural Network with three different encodings, including dictionary, character sequence and bag-of-N-gram encodings. The dictionary encoding method provides the best performance, as lexical constraints improve precision by pruning invalid word detections.

For the 2013 ICDAR Robust Reading task, USTB_TexStar developed by Yin et al. [25] obtained 1st place [26, 27]. This system introduces an MSER pruning algorithm to improve precision, with single-link clustering to group candidate regions instead of empirically selecting a threshold, and character classification used to filter non-text candidates in the final step. In second place, Neumann et al. used an MSER-based algorithm [15]. In [16], they compared filtering methods considering isolated CCs, CC pairs, and CC triples, and find the additional context available in features extracted from CC triples provide the best performance. In follow-on work [17] they propose cascaded filtering of MSER regions. Simple features are computed at first to remove easily detected backgrounds, after which more complex features are used with an AdaBoost classifier. In [18], multiple recognition and character sequence candidates are used to improve recall, along with improved handling for varying character sizes using a Gaussian scale-space pyramid.

In our work we use a modified Convolutional k-means-based sliding window detection performed in two passes, using first a coarse resolution, and then regions of interest with higher resolution matching and variations in the rotation and aspect ratio of the detection window. We consider multiple scales using direct subsampling of the input image, to which we apply convolution masks for detection. Character hypotheses are formed from the 'fine' (higher resolution) detectors, and then regions are grown based on color gradients, with multiple edge hypotheses used to generate character candidates which are then validated. Characters are then both merged into words and validated again in a final pass, using the complete graph over detected characters. Details of our approach are described in the next section.

3. Methodology

An illustration of our system is shown in Figure 1, and example outputs from each stage are shown in Figure 2. The system is a cascade, with each step designed to address a specific aspect of natural scene text, initially generating and then validating hypotheses passed on to the next step. Similar to a design strategy used in the Viola-Jones face detector [21], recall in the coarse detector, fine detector and region growing steps is kept high by choosing a threshold producing a fixed level of recall in the training data (possibly sacrificing precision to some degree), in order to avoid

[2] http://www.stradvision.com/

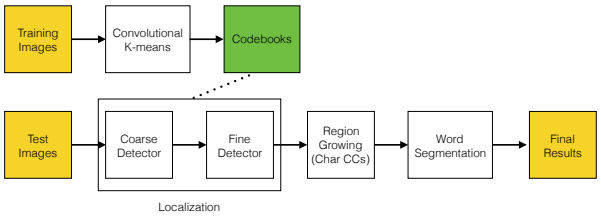


Figure 1: System Architecture. The main stages localize text pixels, generate and verify connected components as characters, and finally segment words.

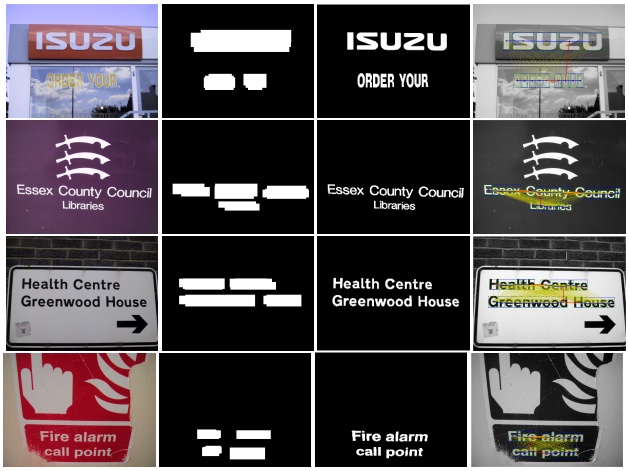


Figure 2: Detection Examples. From left-to-right inputs are shown followed by their results for: i) coarse detection, ii) fine detection and region growing, and iii) word segmentation. In word detection results, words are in blue boxes, and characters in green boxes. For simplicity, we represent word segmentation results using a Minimum Spanning Tree (MST) over character pairs (yellow lines) defining character merges (blue lines) and word separations (red lines). In actual fact, all character pairs are labeled as 'merge' or 'split.'

false negatives.

3.1. Text-Conv Detection System

We propose a feature learning-based convolutional detector called Text-Conv. It is composed of a coarse-to-fine two step scanning scheme, mimicking glancing and focused attention by the human visual system.

3.1.1 Feature Learning Using Convolutional K-means

Feature learning algorithms were developed using restricted Boltzmann machines (rBM) [6] or auto-encoders [20], etc. However, these algorithms are computationally expensive and not suitable for large images or real-time applications. Coates et al. [5] proposed the convolutional k-means feature learning algorithm, using simple k-means clustering to learn feature banks. Convolutional k-means considers the angular distance between training sample patches, and generates cluster center vectors (i.e. convolution masks) through iterative learning. The cluster centers represent typical patterns, including horizontal and vertical bars, corners, sloping bars, zebra textures etc. These patterns are learned automatically from data, without elaborate modeling. The only hyperparameter that needs tuning is the number of clusters. These learned features are very similar to those acquired by an auto-encoder or rBM, and lead to very similar performance [6].

For sample matrix X containing m samples with n features, we make the matrix size $n \times m$. Each column is a sample vector, and each row corresponds to a feature, $X \in \mathbb{R}^{n \times m}$. For initialization, we randomly pick k samples (initial cluster centers) from the sample matrix X and then normalize each vector, so cluster center matrix $D \in \mathbb{R}^{n \times k}$. Our goal is to minimize Equation 1 (see Coates et al. [5]):

$$\sum_i ||Ds_i - x_i||^2 \quad (1)$$

Each column of D is the normalized basis vector. s_i is a bit vector (*hot encoding*) with exactly one non-zero element representing the cluster center (column of D) training sample x_i belongs to. Its magnitude is the dot product between the sample and closest cluster center. To find a matrix D that minimizes the total distance from samples to cluster centers, we alternatively minimize D and s_i.

In our experiments, $k = 1000$ convolution masks (cluster centers) are learned for both the coarse and fine detectors. We found empirically that fewer than 1000 masks reduce accuracy, while additional masks lead to only minor improvements in accuracy.

3.1.2 Coarse-to-Fine Character Detection

Conventionally in a CNN, the system is trained using algorithms such as back-propagation. Instead, we use confidence-rated AdaBoost to classify patches as foreground (text) and background (non-text). Unlike the original AdaBoost which provides discrete labels in $\{-1, 1\}$, a confidence-weighted AdaBoost classifier produces both a label and confidence value. Applying our detector to windows across the image, we obtain a detection hotmap (saliency map). This is computationally very expensive. To reduce computation, we implement a coarse-to-fine scan.

The Text-Conv system is trained and applied to testing images after color Sobel edge detection has been applied. Edge images are used so that the influence of luminance inhomogeneity can be reduced. Our coarse-to-fine scanning divides the raster scanning patch generation and classification into two stages. In the coarse stage, the image is scanned using a larger step size, i.e. with lower resolution

but faster execution. The saliency map of the coarse detector will then be used as a reference for fine detection. In the fine scan, only regions of interest found by the coarse detector are considered. The fine scan uses a very small step size (1 pixel) to ensure high recall.

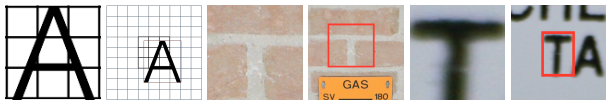
The coarse detector patch size is 32 by 32 pixels, with a step size of 16 pixels. Therefore, two consecutive patches have 50% area overlap. We found that using a standard window such as shown in Figure 3(a) to capture local information gives poor performance. However, neighboring pixels provide discriminative information (see Figure 3 (c)-(f)). To consider neighboring patches during detection, we design the image patch as shown in Figure 3(b). The center block contains a 3×3 grid containing the target region at center. The eight neighboring blocks around the center block provide contextual information.

Coarse detection produces a hotmap representing the likelihood of text. An example of a coarse detection hotmap with and without contextual features is shown in Figure 4. As seen in the example, a substantial increase in discriminative power is provided by the surrounding blocks. To define regions of interest for the fine detector, the hotmap is thresholded. The threshold is defined as the value obtaining the highest f-measure on a validation sample taken from the training data, as shown in Figure 5(a).

For fine-grained character detection, the scan step size is reduced to 1 pixel, and so it is less important to consider partial overlap with characters, and surrounding pixels are ignored. The fine detector is trained using patches containing fully overlapped characters as foreground. To handle different text aspect ratios caused by perspective transformations and improve detection for very narrow and wide characters, we compute multiple window aspect ratios and image rotation angles. A grid search is performed over aspect ratios and rotational angles, with output values maximal pooled. We compute aspect ratios from 0.6 to 1.4, using step size 0.2. We also consider small rotations from -6° to 6°, using a step size of 2°.

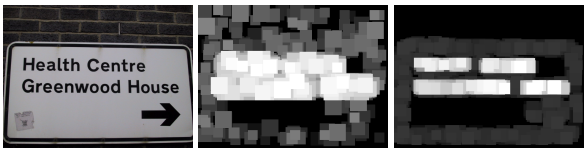
These transformations and a smaller scan step size make fine detection much more computationally expensive. However, these computations are performed only in regions of interest. The fine hotmap is then thresholded to maximize the f-measure (see Figure 5(b)). Surviving pixels provide seeds for the subsequent region growing step.

Scales. In order to catch texts in different sizes, the coarse detector considers multiple scales. However, for fine detection, only scales containing regions of interest remaining after thresholding the coarse detection are considered, along with the next-largest and next-smallest scales. We iteratively decrease the image size by 10%, obtaining 30 different scales. The detector will cover texts with a size variation of about 23.59 times. The definition of character bounding box overlap for foreground patches is based upon



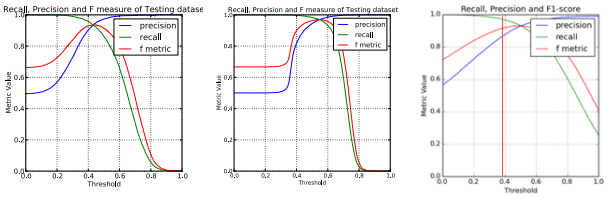
(a) (b) (c) (d) (e) (f)

Figure 3: Local 3x3 (a) and Contextual 9x9 (b) detection windows. Contextual windows are a standard 3x3 window surrounded by features from an 8 neighborhood. Panels (c)-(f) illustrate resolving ambiguous local features by context.



(a) Input (b) Local 3x3 (c) Contextual 9x9

Figure 4: For the ICDAR 2015 test image in (a), differences in coarse character detection maps for a standard 3x3 sliding window (b) vs. a contextual 9x9 widow (c) are shown.



(a) Coarse Detection (b) Fine Detection (c) Verification

Figure 5: Recall, Precision and F-Measures at Different Classification Confidence Thresholds for (a) Coarse Detection, (b) Fine Detection and (c) Verification. Training image samples are divided into 80% training and 20% validation. Validation set results are shown.

the scanning step size and scale ratio. If a patch is less than 10% smaller in width or height of a character bounding box, and the overlapping area is greater than $0.75^2 = 0.56$, then it is considered a foreground patch. This defines a *minimal target overlap*, to help insure detections only when a substantial portion of a character is seen in the detection window.

3.2. Region Growing

The thresholded fine detection saliency map provides seeds for a flood-filling type of region growing, in order to form CCs. For each position that was classified as text from Text-Conv, we start to grow regions iteratively until they reach an edge or a large shift in color. Some small false CCs might appear, due to small homogeneous regions around character edges. We implement a surrounding suppression technique to remove these easy negative regions. When a text region is detected, its surrounding area is suppressed. The surrounding area is defined as 5 pixels in our

experiments.

Consider image I as a mapping: $D \in Z^2 \rightarrow S$. Region growing will add new pixels into foreground regions iteratively by considering two criteria: 1) the edge intensity along the growing direction is small, and 2) the color difference between a newly added pixel and a region seed pixel is small.

If $q = (i_q, j_q)$ is a pixel immediately adjacent to the contour of CC Q (while not in Q), and $p = (i_p, j_p)$ a pixel in Q, then the growing direction is defined as $\Theta = atan2(i_q - i_p, j_q - j_p)$. If multiple pixels $p_1, p_2, ..., p_n \in Q$ in D are adjacent to q, the growing direction is defined as the average of all directions for $p_1, p_2, ..., p_n \in Q$.

Edge images are computed from the intensity gradient in each color channel, in the horizontal and vertical directions. For three color channels R, G, B, the color gradient map of the image can be computed by the Laplacian Matrix L:

$$L = D^T D, \text{where } D = \begin{bmatrix} \frac{\partial R}{\partial x} & \frac{\partial R}{\partial y} \\ \frac{\partial G}{\partial x} & \frac{\partial G}{\partial y} \\ \frac{\partial B}{\partial x} & \frac{\partial B}{\partial y} \end{bmatrix} \quad (2)$$

The gradient amplitude can be computed using the largest eigenvalues of the Laplacian matrix λ, and gradient direction can be computed using the eigenvector corresponding to λ. Intensities of the gradient, $\frac{\partial R}{\partial x}$ etc. are computed discretely using Sobel kernels. The angle between the growing direction and gradient direction is computed by $\delta\Theta = \Theta_e - \Theta_g$, where Θ_e is the gradient (edge) direction and Θ_g is the growing direction. Notice that $\delta\Theta$ is regularized, therefore its range is between $(-\pi/2, \pi/2)$. The region growing criterion C is as follows:

$$C = cos(\delta\Theta)\ \lambda + \frac{\sum_{c \in R,G,B}(|I_{c,q} - I_{c,seed}|)}{Z} \quad (3)$$

where the first term represents the edge intensity along the growing direction, and the second term represents the color difference between boundary pixel q and the region's seed pixel. Z is a normalization factor. Regions grow from seeds iteratively, adding valid boundary pixels into the foreground region based on C.

Initially, seed pixels and region boundaries are labeled as foreground/background respectively. Unlabeled pixels with minimal cost are labeled iteratively. Region growing stops when no unlabeled pixels exist between foreground and background.

Validating Character CCs. After growing candidate CCs for characters, an AdaBoost classifier is trained to prune CCs that are invalid. To accommodate the high variation in colors and intensities in natural scenes, using the input image we generate multiple Canny edge maps, by using multiple Gaussian smoothing kernels with sizes from 3 to 11 pixels, with variance equal to half the kernel size. Edge point thresholds are defined from 50% up to 90% of the maximum gradient value.

We train the verification classifier using the pixel level ground truth provided in the ICDAR data set. We count overlapping pixels between generated CCs and true characters, and use CCs whose area overlapping area is greater than 90% as foreground. The fine-grained AdaBoost detector used for Text-Conv is applied. Precision and recall values can be tuned by choosing different cut-off thresholds. As we need to keep as many true positives as possible for later processing, we select the threshold so that recall is higher than 95% (see Figure 5(c)). A similar thresholding technique was used in the Viola-Jones face detector [21].

In a cascaded system, hypotheses are eliminated stage by stage. To keep final recall within a reasonable range, recall in each stage should be kept high. However, precision in each stage may be relatively low. As false positive samples are filtered in cascaded stages, the final precision of the system may also be high.

3.3. Word-Graph

Characters within a word usually have similar color and size, with relatively small and equally distributed distances. For English, natural scene text usually appears in horizontal text lines with small rotation, with some exceptions. For example, isolated characters may be a word (e.g. 'a'), and text lines might not be straight, or may be curved. Objects with regular textures like windows and bricks share some patterns with text, making it difficult for them to be removed using spatial relationship information alone. And so, to reliably merge characters into words, one needs to group CCs based on both their appearance and spatial relationships.

The Word-Graph algorithm groups detected character CCs into words, and then uses context to prune false positive CCs. It uses a graph model $G(V, E)$, where characters are vertices V and their relationships are edges E. Two Random Forest classifiers are used in Word-Graph; the first for character merge/split classification, and the second filters invalid characters after forming words. Instead of designing features and predefined thresholds for linking characters as in [23], Word-Graph is data-driven, with little human intervention.

Twenty-nine features are defined for Word-Graph edges, including greyscale intensity differences and seven bounding box features: three centroid distances (horizontal, vertical, and Euclidean), the smallest bounding box vertical or horizontal distance, differences in width and height, and the angle of the main axis orientation (via PCA). Raw bounding box features are used along with three normalizations (by minimum, maximum, and mean). For example, minimum normalization of center distance in the x direction for

CCs i and j is given by:

$$\frac{|x_{c,i} - x_{c,j}|}{min(width_i, width_j)} \quad (4)$$

When merging characters into words, challenges include distant characters belonging to the same word, and adjacent characters belonging to separate words. One strategy for tackling this problem is to examine only neighboring characters for word segmentation. In the training data, a minimum spanning tree based on spatial distance is used to select training edges. This insures that only edges between neighboring characters are used to define positive (merge) and negative (split) examples, increasing training speed, and increasing the separability of the classes (vs. using all pairs of characters).

To increase recall, words are located by applying the Random Forest classifier to *all* edges in the complete graph over detected character CCs. A transitive relation over 'merge' edges is then used to locate words.

Second Stage Character Validation. A second random forest is used to remove spurious 'characters' from the word graph. Visual features for characters are combined with features on edges (see above) connected to a character in the word graph. CCs are characterized by 900 convolutional features generated on a 3×3 spatial pooling grid using 100 codebooks learned by convolutional k-means. Along with this, we include features from the highest and lowest confidence edges, along with average feature values for 1) all connected edges, and 2) connected MST edges. Characters from ground truth are used to define complete and MST graphs over characters, and these ideal graphs are then used in training the random forest.

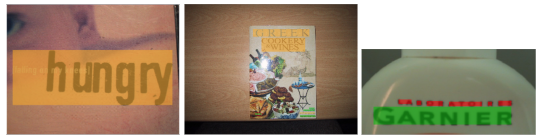
4. Experiments

We tested our system using the ICDAR 2015 Focused Scene Text dataset (Challenge 2, Task 1), containing 258 training images and 251 testing images. The evaluation metric checks the overlapping area for each word detection with ground truth and computes the final precision, recall and f-measure based on the total number of words that are correctly detected. To be considered a valid detection, the bounding box of a word hypothesis must have a precision of 40% and a recall of 80%. Many-to-one and one-to-many matching is implemented to accommodate merged and split words, using a 20% accuracy scaling as a penalty [12].

4.1. Training

Coarse-to-Fine Character Pixel Detection. The set of 1000 cluster centers (convolution masks) obtained via Convolutional k-means are learned from training images, with each feature 8×8 in size. These features are then convolved with the image to form 32×32 patches. For coarse detection, each 32×32 patch is spatial pooled into 3×3 grids.

Figure 6: Correct Word Detections.



(a) (b) (c)

(d) (e) (f)

Figure 7: Word Detection Errors. Red boxes are false positives, yellow boxes are under-segmented words, and blue boxes are over-segmented words.

Including context blocks, 9×9 grids are generated. In total, 72588 (36294 foreground and 36294 background) training samples are generated using ICDAR training images for coarse detection. To train the fine detector, 50,000 randomly selected foreground images from Wang's synthetic dataset [22] are used, along with 50,000 randomly selected background samples generated from ICDAR images. The AdaBoost classifier is trained towards minimum error, but by utilizing the confidence-rated classifier, we can select a final threshold confidence value to maximize the f-measure. This is crucial when foreground and background samples are highly unbalanced. Precision, recall and f-measures are provided in Figure 5a and 5b, where we pick the maximal f-measure point as the threshold to finally cut the foreground regions. Examples of coarse detection after thresholding are shown in Figure 2 column 2.

Comparing the hotmap we have generated to Coates' et al.'s results for the ICDAR 2013 dataset [5], we have achieved higher performance in patch level in terms of Area Under Curve of Precision / Recall. Our detection hotmap has 71.2% AUC, while Coates et. al obtain 62%.

Region Growing. In total, 4721 foreground and 7835 background samples are used for training. To tune the final threshold for the confidence-rated classifier, we focus on higher recall rather than precision. The highest f-measure point for this classifier has a recall value less than 95% on our validation set. This is lower than we would prefer, and so we set the threshold to the point where we obtain 96% recall (see Figure 5c). Experiments show that this configu-

ration produces the highest accuracy for the entire system. Some examples of region growing results after verification are shown in Figure 2 column 3.

Word-Graph Figure 2 column 4 shows examples of edge classification results for segmentation. To train the Random Forest classifier for edges, an important issue is that the foreground and background training samples are unbalanced. Experiments show that balancing samples is very important for reliable segmentation and character verification (increasing f-measure from 95% without balancing to 99% with balancing). As non-neighboring edges are removed from the training set, within-word edge samples are far fewer than between-word samples. To deal with the class imbalance, we considered different sampling methods, including using the original distribution, over-sampling to balance the classes (using SMOTE [4]) and random subsampling. Surprisingly, we found that subsampling could produce better accuracy for testing samples than over-sampling. We were able to train an accurate segmenter using just 4557 foreground and 4557 background samples. 929 features are used to train the random forest with 100 trees and a maximum depth of 10, and each node split considering $\sqrt{929} \approx 30$ features.

After Word-Graph has completed our text detector has finished, and word bounding box coordinates are extracted and saved into a text file.

4.2. Results

We evaluated our system using the ICDAR 2015 Robust Reading evaluation website, by uploading our word coordinate files onto their system. Examples of correct detections are shown in Figure 6. The online evaluation system checks the overlapping area for each detection with ground truth and computes the final precision, recall and f-measure based on the total number of words that are correctly detected. In the competition, systems were compared based on their word detection results; our system obtained stronger recall, precision and f-measure than the winning system from StradVision Corporation, as shown in Table 1.

It is worth noting that including differing rotations and aspect ratios during fine-grained character detection had a dramatic effect on accuracy. Without these, the f-measure for word detection decreases from 86.77% to 64.72%.

Although our system achieved state-of-the-art performance for word detection, there is still room for improvement. The system failed to deal with some overlapped text lines properly, missing words in some images as shown in Figure 7a and 7b. Isolated characters are more likely to be missed by our detector. Some words have large within-word distance between characters (see Figure 7c). Some specular highlights wash out characters, and there is no way to retrieve the information using image data alone, as in Figure 7d. Some handwritten characters are also missed due to lack of training data, in Figure 7e. In some cases, we found valid words, but they were recognized as false positives. For example in Figure 7f, the red regions contain digits (0/1), but they are considered to be background decorations of the book cover in ground truth.

Table 1: ICDAR 2015 Focused Scene Text Results.

	Recall (%)	Precision (%)	F-Score (%)
StradV. (Word-bb) [12]	80.15	90.93	85.20
Our System			
Word-bb	**81.02**	**93.39**	**86.77**
Char-bb	87.51	93.20	90.26
Pixels	92.75	93.53	93.14

Table 2: Mean Execution Time (seconds/image). System: Intel Xenon CPU w. 24 processors (2.93GHz), 96GB RAM, GeForce GTX 480 w. 1GB GPU memory.

	CPU	GPU
Convolution	503.4	**77.3**
Coarse Detection	10.9	**2.7**
Fine Detection	744.2	**55.7**
Region Growing	15.1	**10.9**
Word Seg.	**2.3**	2.611
TOTAL	1275.9	**149.3**

As seen in Table 1, character bounding box level accuracy is higher than word bounding box accuracy (f-measure of 90.26%), suggesting that word segmentation rates may be increased through improving Word-Graph. Pixel level accuracy is even higher, with an f-measure greater than 93%. As seen in Figure 2, our system often creates pixel-accurate masks for characters, even in complex scenes.

Our system is implemented in Python. As seen in Table 2, convolutional feature generation and fine detection consume most of the execution time. This is because we apply convolution at different scales and use a grid search over aspect ratios and rotation angles for fine detection, requiring a very large number of convolutions. Convolution and AdaBoost classification may both be accelerated utilizing a GPU. Using Theano [2, 3], average execution time is reduced from about 20 minutes to 2.5 minutes.

5. Conclusion

We have proposed a relatively simple cascaded text detection system that is accurate at the pixel, character and word levels, and produces state-of-the-art performance on a challenging dataset. Contextual features, a coarse-to-fine detection strategy, and using greater visual detail to define targets in later stages help improve sliding window-based character detection. Character detection is cascaded with multiple validation steps, culminating in detected words providing contextual constraints at the final detection stage.

Faster execution can be obtained by re-implementing in C and using multiple GPUs or dedicated hardware for convolution. Over-segmenting input images into 'super-pixels'

based on color and edge information could also significantly reduce the number of detection windows we need to consider, at which point the system might even run in real-time.

It is important to note that we obtain high accuracy using only visual features, without the use of a language model or dictionary. However, detection can probably be improved by integrating character recognition and models for a specified language.

Acknowledgements. We thank Kenny Davila and Ben Miller-Jacobson for providing helpful feedback. This material is based upon work supported by the National Science Foundation (USA) under Grant Nos. IIS-1016815 and HCC-1218801.

References

[1] B. Bai, F. Yin, and C. L. Liu. Scene text localization using gradient local correlation. In *Int'l Conf. Document Analysis and Recognition (ICDAR)*, pages 1380–1384, Aug 2013.

[2] F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. J. Goodfellow, A. Bergeron, N. Bouchard, and Y. Bengio. Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning Workshop (NIPS), Dec. 2012. (CoRR abs/1211/5590).

[3] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio. Theano: a CPU and GPU math expression compiler. In *Proc. Python for Scientific Computing Conference (SciPy)*, June 2010.

[4] N. Chawla, K. Bowyer, L. Hall, and W. Kegelmeyer. SMOTE: synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, 2002.

[5] A. Coates, B. Carpenter, C. Case, S. Satheesh, B. Suresh, T. Wang, D. Wu, and A. Ng. Text detection and character recognition in scene images with unsupervised feature learning. In *Int'l Conf. on Document Analysis and Recognition (ICDAR)*, pages 440–445, Sept 2011.

[6] A. Coates, A. Y. Ng, and H. Lee. An analysis of single-layer networks in unsupervised feature learning. In *Int'l Conf. Artificial Intelligence and Statistics (AISTATS)*, pages 215–223, 2011.

[7] B. Epshtein, E. Ofek, and Y. Wexler. Detecting text in natural scenes with stroke width transform. In *Proc. CVPR*, pages 2963–2970, June 2010.

[8] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 32(9):1627–1645, Sept 2010.

[9] P. Felzenszwalb and D. Huttenlocher. Efficient belief propagation for early vision. In *Proc. CVPR*, volume 1, pages I–261–I–268, June 2004.

[10] T. He, W. Huang, Y. Qiao, and J. Yao. Text-Attentional Convolutional Neural Networks for Scene Text Detection. *arXiv:1510.03283*, Oct. 2015.

[11] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman. Synthetic data and artificial neural networks for natural scene text recognition. *CoRR*, abs/1406.2227, 2014.

[12] D. Karatzas, L. Gomez-Bigorda, A. Nicolaou, S. Ghosh, A. Bagdanov, M. Iwamura, J. Matas, L. Neumann, V. R. Chandrasekhar, S. Lu, et al. ICDAR 2015 competition on robust reading. *Int'l Conf. Document Analysis and Recognition (ICDAR)*, pages 1156–1160, 2015.

[13] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov 1998.

[14] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide-baseline stereo from maximally stable extremal regions. *Image and Vision Computing*, 22(10):761–767, 2004.

[15] L. Neumann and J. Matas. A method for text localization and recognition in real-world images. In *Proc. Asian Conf. Computer Vision (ACCV)*, pages 770–783. Springer, 2011.

[16] L. Neumann and J. Matas. Text localization in real-world images using efficiently pruned exhaustive search. In *Int'l Conf. Document Analysis and Recognition (ICDAR)*, pages 687–691, Sept 2011.

[17] L. Neumann and J. Matas. Real-time scene text localization and recognition. In *Proc. CVPR*, pages 3538–3545, June 2012.

[18] L. Neumann and J. Matas. On combining multiple segmentations in scene text recognition. In *Int'l Conf. Document Analysis and Recognition (ICDAR)*, pages 523–527, Aug 2013.

[19] Y.-F. Pan, X. Hou, and C.-L. Liu. Text localization in natural scene images based on conditional random field. In *Int'l Conf. Document Analysis and Recognition (ICDAR)*, pages 6–10, July 2009.

[20] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *J. Mach. Learn. Res.*, 11:3371–3408, Dec. 2010.

[21] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proc. CVPR*, pages I–511–I–518, Dec. 2001.

[22] T. Wang, D. Wu, A. Coates, and A. Ng. End-to-end text recognition with convolutional neural networks. In *Int' l Conf. Pattern Recognition (ICPR)*, pages 3304–3308, Nov 2012.

[23] C. Yao, X. Bai, W. Liu, Y. Ma, and Z. Tu. Detecting texts of arbitrary orientations in natural images. In *Proc. CVPR*, pages 1083–1090, June 2012.

[24] Q. Ye and D. Doermann. Text detection and recognition in imagery: A survey. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 37(7):1480–1500, July 2015.

[25] X. Yin, X.-C. Yin, H.-W. Hao, and K. Iqbal. Effective text localization in natural scene images with mser, geometry-based grouping and adaboost. In *Int'l Conf. Pattern Recognition (ICPR)*, pages 725–728, Nov 2012.

[26] X.-C. Yin, X. Yin, K. Huang, and H.-W. Hao. Accurate and robust text detection: A step-in for text retrieval in natural scene images. In *ACM Conf. Research and Development in Information Retrieval (SIGIR)*, pages 1091–1092, July 2013.

[27] X.-C. Yin, X. Yin, K. Huang, and H.-W. Hao. Robust text detection in natural scene images. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 36(5):970–983, May 2014.