A Fast and Robust Text Spotter

Siyang Qin Roberto Manduchi
Computer Engineering Department, University of California, Santa Cruz
siqin@soe.ucsc.edu; manduchi@soe.ucsc.edu

Abstract

We introduce an algorithm for text detection and localization ("spotting") that is computationally efficient and produces state-of-the-art results. Our system uses multi-channel MSERs to detect a large number of promising regions, then subsamples these regions using a clustering approach. Representatives of region clusters are binarized and then passed on to a deep network. A final line grouping stage forms word-level segments. On the ICDAR 2011 and 2015 benchmarks, our algorithm obtains an F-score of 82% and 83%, respectively, at a computational cost of 1.2 seconds per frame. We also introduce a version that is three times as fast, with only a slight reduction in performance.

1. Introduction

Systems that can automatically detect the presence of text in an image (text spotters) may find application in multiple practical scenarios, such as video surveillance, forensic, video annotation, mobile OCR. Our main interest in text spotting stems from its potential application as an assistive device for blind people. Being able to detect and read visible text (e.g., a name tag at a door, a wayfinding sign in an airport, the name of a store posted above its entrance) could provide a blind traveler with enhanced environment awareness and better self-confidence. By supporting independent travel, this technology could have direct consequences in terms of education, employment, socialization and recreation for the visually impaired community.

To access text, blind users could rely on the camera of their hand-held smartphone, or on a wearable camera that is tethered or wirelessly connected to the smartphone. It is important to note that the task of detecting and reading a posted piece of text is in fact a cooperation between the user who is maneuvering the camera, and the system, which provides feedback to the user (for example, via acoustic interface or synthetic speech). In a typical situation, the user would first detect the presence of text (perhaps serendipitously, or after intentionally exploring the scene with the camera). It is not critical that the system be able to read the text at this point: in fact, the image could have low resolution if the text is seen from a distance, or the text could be only partially framed, making reading difficult or impossible. Upon being informed by the system that text has been detected, the user could move closer to it and try to take a well-framed, well-resolved snapshot, which could then be processed by an actual text reader (OCR). Non-visual interfaces could be used to help the blind user in this task. This operation trades reading accuracy for redundancy: as long as the user is able to keep the text within view of the camera, and the text spotter can reliably detect and localize text in the video frames, a large number of images containing the text are available for OCR, maximizing the chance that at least one of these images can be read correctly.

The system presented in this contribution focuses solely on detecting and localizing text in an image, deferring reading to later in the pipeline. High priority was given to efficiency (computational speed) and of course performance (as measured by standard criteria [23]). At a speed of 1.2 seconds per image (VGA size), our text spotter achieves F-scores of 82% and 83% on the ICDAR 2011 and 2015 benchmarks, respectively. For comparison, the published algorithm with best reported results [27] achieves an F-score of 0.80 on both data sets at much lower speed. A faster version of our algorithm (running at 0.38 seconds per image) results in F-scores of 0.80 and 0.79, respectively.

The general structure of our algorithm, shown in Fig. 1 is similar to that of other successful systems presented in the literature, with some important differences. We first process the input color image with multi-channel MSER [14], using a very conservative threshold. We then carefully prune the ensemble of resulting extremal regions; this is a critical step to reduce the computational cost of all subsequent modules. Our pruning procedure is an original contribution of this paper. The remaining MSERs are then resized to fit 32×32 bounding boxes, and fed to a convolutional neural network (CNN). Unlike standard approaches that use the grayscale-valued array as input, in our work the input to the CNN is a binary indicator mask (see Fig. 2). Our experimental results show that this simple "trick" can increase classification accuracy significantly. Rather than training the CNN

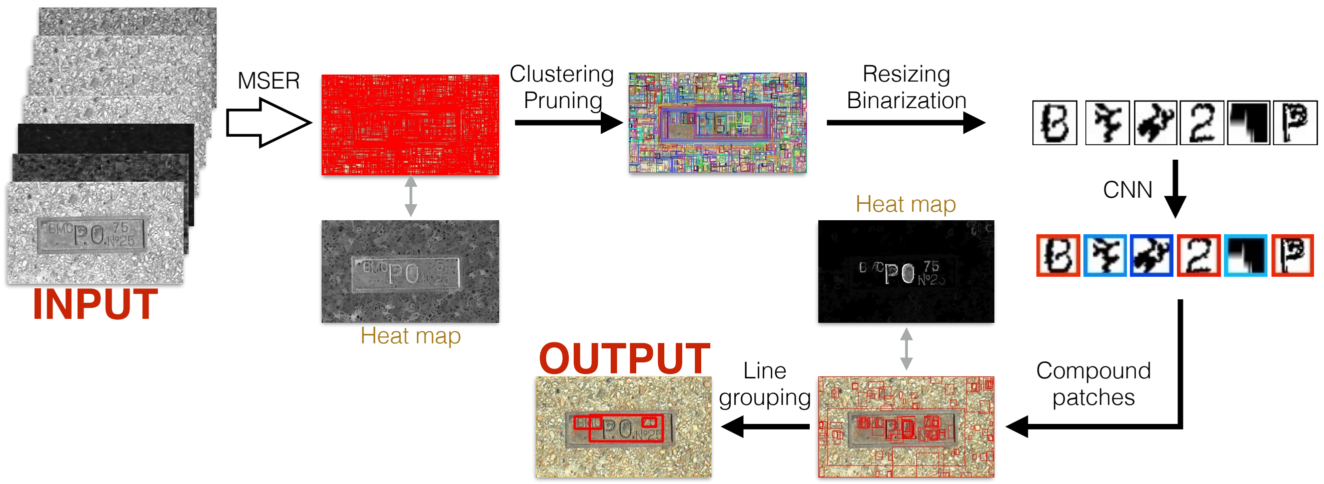


Figure 1. The overall structure of our algorithm. Multichannel MSERs (Stage I) are thinned out through clustering and pruning (Stage II), then resized and binarized before being fed to a CNN classifier (Stage III - classification results are encoded by the color of the frame around each patch.) The compound patches that have been classified positively are then grouped into text lines (Stage IV).

on individual characters (synthetically produced or manually segmented), as is typical for similar algorithms, we mine positive and negative examples from training data sets that are only labeled at the word level. The resulting positive examples may contain digrams and sometimes other n-grams; this is not a problem, as our goal is word-level detection and labeling, rather than individual character recognition. Finally, the score assigned by CNN to the individual regions is used to guide a simple but effective line grouping algorithm. Calibration of the individual system components is performed using a metric of precision/recall that is specifically designed for text spotting.

2. Related Work

Text detection and localization methods have been traditionally grouped in two main categories. The first category contains sliding window approaches [22, 9, 27, 10, 2, 6, 21, 10]. Sliding window analysis has been a cornerstone of computer vision since its infancy. It is a well understood technique with remarkable robustness to noise and to undesired effects such as disconnected strokes. Unfortunately, its computational cost is usually high, considering that multiple size windows are normally needed.

Methods belonging to the second group extract candidate text characters based on local characteristics. Stroke width transform (SWT) [5] and maximally stable extremal regions (MSER) [14, 19] are among the most popular approaches. Such methods allow one to concentrate only on the more promising candidate regions, but are quite sensitive to noise and blur. SWT finds character candidates by grouping pixels with similar stroke width into connected components, where stroke widths are computed from almost parallel edges. MSER, an universal tool used in multiple applications of computer vision, has also been shown to work well for the task of identifying text characters. For example, Neumann and Matas [15] proposed a method to perform efficient sequential selection from the set of extremal regions in the image to obtain character candidates. Huang *et al.* [7] introduced the Stroke Feature Transform, which extends the Stroke Width Transform idea by considering color for increased robustness, along with two novel Text Covariance Descriptors used to train a classifier. Chen *et al.* [1] enhanced the MSER algorithm by adding Canny edge cues, with the goal of increased robustness to blur and noise.

Some works attempt to combine the advantages of sliding-window and of connected component methods. Neumann and Matas [17] proposed a novel approach to character detection. An image region is treated as candidate character if it contains strokes of specific orientations in a specific relative position. The number of candidate character regions is reduced by three orders of magnitude respect to traditional sliding-window approaches. Zamberletti *et al.* [26] proposed a hybrid system that generates a text confidence map using a sliding-window classifier based on fast feature pyramid [4], then remove false positives using MSERs.

Recent years have witnessed tremendous progress in unsupervised feature discovery and deep learning; these techniques have been applied to almost every area of computer vision, and scene text localization is no exception. Rather than rely on hand-designed features, deep convolutional neural networks (CNN) [13] use hierarchical and over-complete features learned from large training data sets. Use of CNN has enabled substantial improvement in text detection and localization accuracy. For example, Wang *et* *al.* [22] use a sliding window to extract candidate regions that are then fed to a CNN (with the features of its first convolutional layer trained in an unsupervised manner [3]). Huang *et al.* [8] use MSER to find candidate regions that are then classified by a CNN similar to that of [22]. Zhang *et al.* [27] rely on the spatial symmetry that is charateristic of character groups, then use CNN to remove false positives.

For more comprehensive surveys, the reader is referred to [12, 20, 24].

3. Method

As customary for text spotting [2], we structure our algorithm as a cascaded classifier, with the initial stages designed to have high recall rate. Our system (Fig. 1) can be divided into two main components: the first component produces a set of rectangular patch, each weighted by a confidence value of being contained in a text area (Stages I to III); the second component groups these areas into linear chains – tentative "words" (Stage IV).

3.1. Stage I: MSER Segmentation

The first stage of our algorithm is multi-channel MSER segmentation. MSERs [14], along with Stroke Width Transform, is widely used to identify promising regions in modern text spotters. It is particularly suited to discovery of high-contrast regions such as text characters. We employ very conservative parameters for MSER computation: using the terminology of [14], an extremal region $\mathcal{Q}_i$ is determined to be maximally stable if the incremental area ratio $|\mathcal{Q}_{i+1} \setminus \mathcal{Q}_{i-1}|/|\mathcal{Q}_i|$ is smaller than τ ($\tau = 0.25$ in our implementation). We compute MSERs with both polarities (dark on bright and vice-versa) on 7 image channels: R, G, B, grayscale, H, S, V. MSERs that contain fewer than 30 pixels, or with anomalous format ratio (less than 0.3 or larger than 3) are rejected.

While in many cases a character is well segmented by one or more MSERs, in some situations no MSER can be found that correctly encompasses just one full character. For this reason, several authors (e.g. [8]) have proposed techniques that modify or subdivide these regions, with the purpose of localizing individual letters. This may be necessary when the exemplar set used for training contains individual characters (e.g., synthetically generated). We don't make this assumption in our system: positive exemplars are mined directly from the MSER regions, and thus may contain digram or other n-grams. Consequently, we don't need to modify the regions produced by MSER in any way (except for reshaping them to a common 32×32 size in Stage III.)

3.2. Stage II: Region Clustering and Pruning

MSER segmentation produces a large number (on average, 3151 per image on ICDAR benchmarks) of possibly overlapping rectangular image patches. While it would be possible to pass each one of these patch on to the classifier (Stage III), the computational cost would be prohibitive. We thus need a method to weed out the least promising patches.

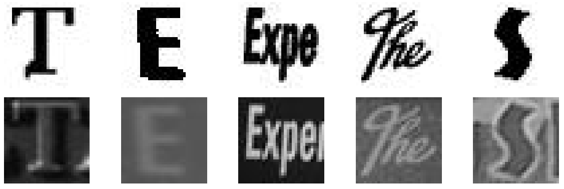


Figure 2. Some resized binary patches fed to CNN, together with their original grayscale counterpart. Note the presence of a trigram and of a four-gram.

Our criterion for clustering and pruning is guided by two empirical observations. The first observation is that actual characters tend to fill their rectangular bounding box[1] more than spurious MSERs. We embody this notion by a simple measure of *fullness* (ratio of the area of the MSER to the area of its rectangular bounding box). Note that other features describing the shape of the MSERs have been used in previous work (e.g.[15]). Measuring these features on all detected MSERs, however, would be computational demanding. The second observation is that multi-channel MSERs tend to cluster around actual text characters, as revealed by observation of the heat map formed the MSERs in most images (see Fig. 1). In this context, a heat map simply assigns a value to each pixel equal to the number of MSERs that overlap on that pixel.

We find clusters of overlapping MSERs' bounding boxes, possibly pruning out small clusters, and extract one cluster representative based on the fullness measure. More precisely, we create a graph with the MSERs as nodes; two nodes are linked by an edge if the Jaccard index between the corresponding MSERs' bounding boxes is larger than 0.8 (where the Jaccard index of two sets is the ratio of the intersection to the union of the sets). Note that this graph can be computed very efficiently using a sweep line and an interval tree. The connected components of this graph are computed. Optionally, connected components with a small number of components can be removed; for example, our "fast" implementation prunes away all clusters containing less than 3 nodes. Finally, the number of MSERs is reduced by selecting one representative MSER per cluster, and specifically the one with highest fullness in the cluster (see Fig. 3). This operation reduces the average number of regions to 1392 per image (without pruning), and to 300 per image (when clusters with less than 3 components are removed).

[1]By "bounding box" of a region we mean the smallest rectangle containing the region with sides pairwise parallel to the image axes .

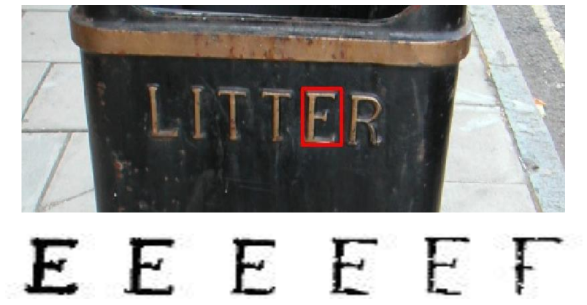


Figure 3. MSERs within a cluster centered at the red rectangle (only a few shown). The MSERs are ordered in decreasing value of fullness. The MSER to the left is chosen as the cluster representative in Stage II.

3.3. Stage III: Region Classification

Patches produced by Stage II are resized to a common square size of 28×28 pixels, then zero-padded to 32×32 pixels squares and fed to a convolutional neural network (CNN). We use a standard CNN structure [22, 8] with two convolutional layers (*conv1* and *conv2*) containing 96 and 256 kernels respectively. Kernels in *conv1* have size of 8×8 pixels; those in *conv2* have size of 2×2 pixels. Each convolutional layer connects to a rectified linear unit (ReLU) and to a max pooling layer. The first pooling layer (*pool1*) performs 5×5 max pooling, the second one (*pool2*) performs 2×2 max pooling. The $2 \times 2 \times 256$ output from the last pooling is passed on to a fully connected layer to obtain a 500 length feature vector which is feed into the SVM classifier, producing the final classification score. Training is fully supervised from exemplars with binary labels.

Unlike other approaches that use character-level exemplars for training (either hand-segmented or synthetically generated), our training samples are obtained with exactly the same method as described in Stage I, from a data set that was hand-segmented at the word level (as available in the ICDAR 2011 and 2015 text localization data sets). We mine positive samples from the training portions of these datas set by first running multi-channel MSER (Stage I, but without the clustering/pruning procedure of Stage II), then treat each resulting rectangular region (bounding box) as a positive sample if (1) this region is substantially contained within a word-labeled rectangle, and (2) its height is similar to the rectangle's height. More precisely, the region must overlap with the word-labeled rectangle by at least 80% of its area, and its height must be between 60% and 120% of the height of the word-labeled rectangle (see Fig. 4). If either condition is unsatisfied, the region is treated as a negative sample. Overall, we obtained approximately 60K positive samples, and seven times as many negative samples, mined from the training portions of the ICDAR 2011 and 2015 text localization data sets. The negative set is then

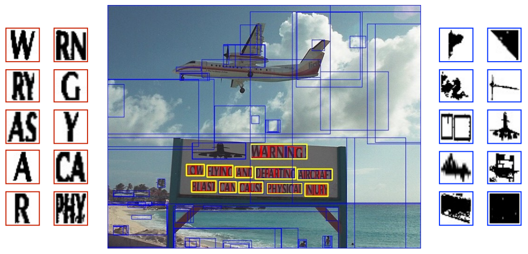


Figure 4. Mining positive (red) and negative (blue) examples from an image that was labeled at word level (yellow rectangles). Only a subset of MSERs are shown on the image for readability.

subsampled to about 110K samples.

The practical importance of not requiring character-level training is twofold. First, it is arguably easier to hand-segment whole words from images, rather than individual characters. Of course, this problem is immaterial if synthetic data sets are created, although the verisimilitude of this synthetic data to real images may be called into question. Second, there is no need to pre-process patches which are suspected to contain multiple characters, an operation that can be challenging and time-consuming.

In a small but significant departure from standard CNN classification approaches, we feed the classifier with a binary image, and precisely with the indicator mask of the MSER in the patch, rather than using the full grayscale value range (Fig. 2). We have found that this simple trick significantly improves results (see Sec.4). While more research is needed to understand the exact reason for this improvement, we speculate that the chosen MSER representatives may overcome the effect of blur and poor contrast, and perhaps remove undesired background clutter that could be otherwise present in the graylevel patch.

3.4. Stage IV: Text Line Grouping

The last stage of our algorithm groups together patches that passed CNN classification into text lines, and separates these groups into "words". As a first step, we recover the connected components of MSERs whose representative was passed on to CNN (Stage II), and create a rectangular bounding box ("compound patch") encompassing all such regions. Each compound patch is assigned the confidence value given by CNN to the corresponding resized representative patch. To find text lines, we resort to a sequential voting strategy with greedy removal. Starting from the compound patch with highest confidence value P_0, we examine all other compound patches; for each other patch P_i, we compute the angles (within $-\pi/2$ and $\pi/2$ from the horizontal direction) of three lines: the line joining the tops of the vertical bisectors of the two patches; the line joining the midpoints of the vertical bisectors; and the line joining their bottoms. We found that all three lines need to be considered, in order to account for rectangle with different sizes. These three angles are then quantized into 36 bins; each line contributes one vote to the corresponding bin's counter, with a weight that is proportional to the CNN classification score P_i, divided by the Euclidean distance between the centers of the two patches. The angle corresponding to the highest bin counter is selected, defining a line drawn from the center of the compound patch. The compound patches intersecting this line are ordered based on their horizontal distance to the patch under consideration, and visited to determine whether they should be added to the current "text line". Two variables, containing the average patch width and height respectively, are updated each time a patch is added to the text line. A strip is centered on the line, with height equal to the current average patch height. A visited patch is added to the line if two conditions are satisfied: (1) the Jaccard index between the patch's vertical bisector and the stripe section aligned with this bisector is larger than 0.5; and (2) the horizontal distance between the visited patch and the last added patch in the same direction (measured at their closest sides) is smaller than twice the average patch width. As soon as a patch is found that does not satisfy condition (2), the process is stopped; all patches assigned to the text line are removed, and the process is started again from the remaining highest confidence compound patch. Finally, line groups with average patch confidence below a certain threshold are removed.

Once a line group is formed, the extent of individual words is determined. Since we don't perform any lexical analysis, this operation is performed solely based on the patches' spatial characteristics. We first compute the average distance between consecutive patches; then, we split the line halfway between any two consecutive patches whose distance is larger than 3 times this value.

4. Experimental Results

Our text spotter was implemented in C++ using OpenCV and the Caffe implementation of CNN [11]. Multichannel MSER was parallelized using OpenMP. The system was benchmarked on a 3.4 GHz, 4 cores desktop with Nvidia Geforce GTX 650 GPU, running Linux. The classifier was trained with the training portion of both the ICDAR 2011 and 2015 data sets.

4.1. Speed

End-to-end computational times for VGA image size are reported in Fig. 5 and Tabs. 1 and 2. Multi-channel MSER takes 50 ms. If all MSERs in the image (3151 on average) are fed into the CNN, a frame is processed in 2.3 s, with 2.1 s used by CNN processing (Stage III). By clustering MSERs and only retaining one cluster representative per cluster (an operation that takes 90 ms), only 1392 patches are sent to CNN on average, reducing the associated processing cost to 980 ms. If clusters with less than 3 MSERs are pruned

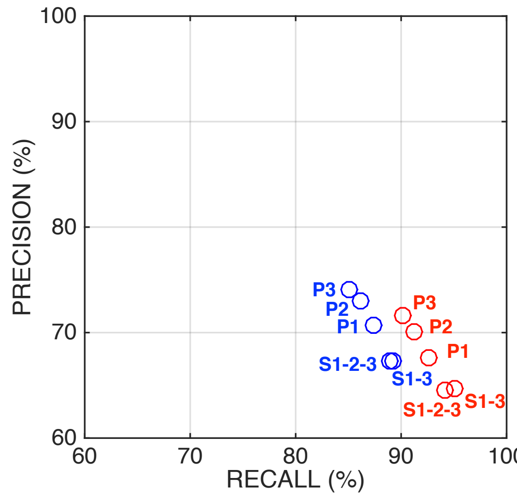


Figure 5. Patch-level evaluation (see Sec. 4.2 for definition of precision/recall in this context). Red: using binary patches. Blue: using greyscale patches. S1-3: Stages I and III with no MSER clustering (end-to-end computational time: 2.3 s/frame). S1-2-3: Stages I, II and III, with one representative per MSER cluster sent to CNN but no cluster pruning (1.2 s/frame). P1, P2, P3: pruning away clusters with less than 1, 2 or 3 MSERs per cluster (550 ms/frame, 380 ms/frame, 320 ms/frame respectively).

away (our "fast" implementation in Tabs. 1 and 2), only 300 patches are sent on to CNN, reducing its cost to 210 ms. Line grouping (Stage IV) takes 30 ms.

4.2. Patch-Level Evaluation

In order to tune the parameters and take design decisions for all initial steps (Stage I–III, before text line grouping), it is important to use a metric that allows for patch-level quality assessment. Unfortunately, the standard word-level metric used in text localization contests [23] would not serve us well for this purpose. We thus devised a simple precision/recall[2] metric that is specifically designed for patch-level evaluation. More precisely, we want to be able to measure how well the characters in the text are covered by patches that are classified positively by our system, as well as to measure how well such patches are contained within word-level segments. Note that use of this metric requires availability of a character-level hand-labeled data set, along with word-level segmentation (we use the ICDAR 2015 text segmentation dataset). Let us emphasize that character-level labels are never used for training; this data set is only used to evaluate patch-level performance.

In our metric, *recall* measures the proportion of ground-truth characters that have been detected. We found the measure of recall defined by Zhange *et al*. [27] appropriate for

[2]Please note that the terms "precision" and "recall" are being overloaded here – they do not have the exact same meaning as the equivalent metrics used in statistics, yet they convey similar meanings.

this metric. Specifically, we assume that a given character has been "detected" if there exists at least one patch classified positively by our system such that at least 80% of the character's bounding box is contained in the patch, and its height no less than 0.7 times and no more than 1.5 times the patch's height. *Precision* measures the proportion of patches that have been correctly classified. We assume that a patch has been "correctly classified" if it is contained for at least 80% inside a word's rectangular region, with the ratio of the patch's height to the word's region height between 0.5 and 1.3.

Fig. 5 shows patch-level precision-recall values for different variants of our algorithm. We would like to highlight the fact that selecting the cluster representatives for CNN classification (rather than classify all MSERs) has a minor effect on precision (at 64.49%) while reducing recall by approximately 1% to 94.10% – but also resulting in an almost twofold increase in speed. As mentioned in Sec. 3.2, the choice of cluster representative is based on the fullness of the MSER's bounding boxes. Indirect evidence of the appropriateness of the fullness measure for representative selection is given by the drop that can be observed (data not shown in the figure) in recall values if the cluster with median fullness or with minimum fullness were to be chosen (93.55% and 88.24%, respectively), with precision remaining stationary (65.03% and 63.07%).

Remarkably, if the full grayscale value is maintained for the pixels in a patch classified by CNN, recall is reduced by as much as 5%, with an almost equivalent increase in precision.

4.3. Word-Level Evaluation

We benchmarked our system with the ICDAR 2011 and 2015 text localization data sets. Comparative results are shown in Tabs. 1 and 2; please note that the definition of precision and recall used for these tests [23] are substantially different from those introduced in the precious section. The slower version of our systems (without cluster pruning; 1.2 s/frame) produces the highest F-score among all other algorithms published to date in both data sets. The faster version (pruning clusters with fewer than 3 MSERs) is three times as fast, with only slightly reduced F-score.

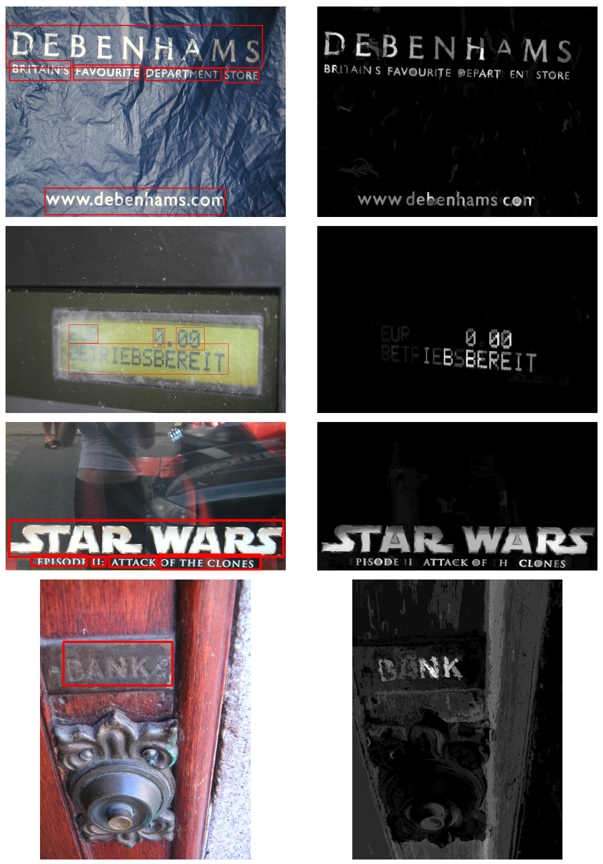
Figs. 6 and 7 show some good and some unsatisfactory results of our algorithm. The figures also show the resulting heat maps, where each pixel is assigned a value equal to the number of MSERs overlapping on that pixel, multiplied by the CNN score assigned to the representative of the cluster at that pixel (when the representative received a positive score). Although we don't use these heat maps directly in our algorithm, they very well represent the relevance of both classification score *and* of the presence of multiple overlapping MSERs as indicators of the presence of text.

	Recall (%)	Precision (%)	F (%)	Time (s/f)
Proposed	77.21	87.59	**82.07**	1.2
Proposed (fast)	72.88	85.76	78.80	**0.38**
Proposed (grayscale)	69.66	85.21	76.65	1.2
Zhang et al [27]	76	84	80	60^*
Huang et al [8]	71	88	78	unknown
Yin et al [25]	68	86	76	0.43
Neumann et al [16]	68	85	75	3.1

Table 1. Word-level benchmarking with the ICDAR 2011 text localization data set [23]. *Proposed* is our method without cluster pruning. *Proposed (fast)* prunes away clusters with less than 3 MSERs in Stage II. *Proposed grayscale* uses grayscale instead of binary patches (no cluster pruning). Computational time is measured in seconds per frame (*refers to a Matlab implementation).

	Recall (%)	Precision (%)	F (%)	Time (s/f)
Proposed	78.67	88.79	**83.42**	1.2
Proposed (fast)	75.18	84.64	79.55	**0.38**
Proposed (grayscale)	71.03	83.45	76.78	1.2
Zhang et al. [27]	74	88	80	60^*
Zamberletti et al [26]	70	86	77	0.75
Neumann et al [18]	72	82	77	0.8
Yin et al. [25]	66	88	76	0.43

Table 2. Word-level benchmarking with the ICDAR 2015 text localization data set [23]. See caption of Tab. 1.


DEBENHAMS
BRITAIN'S FAVOURITE DEPARTMENT STORE
www.debenhams.com
0.00
BETRIEBSBEREIT
STAR WARS
EPISODE II: ATTACK OF THE CLONES
BANK
Figure 6. Some successful results from our algorithm. Left: image with superimposed detected word-level regions. Right: heat map.

5. Conclusions

We have presented an algorithm for text detection and localization that produces state of the art results while be-

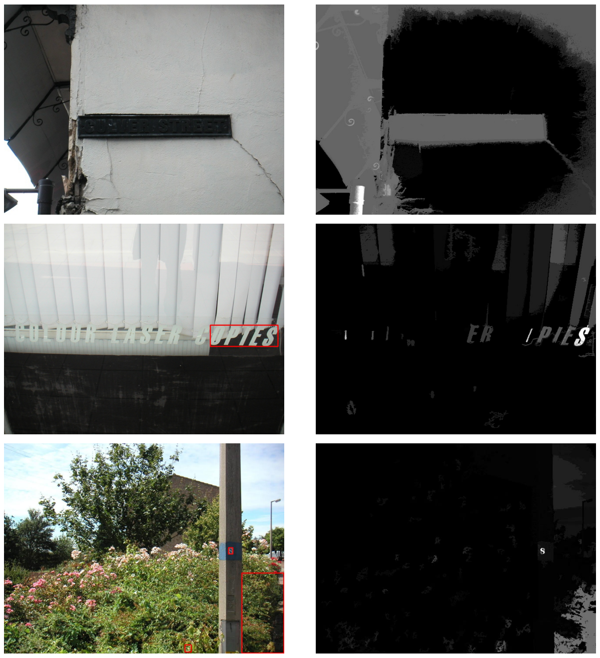


Figure 7. Examples of misdetections. Left: image with superimposed detected word-level regions. Right: heat map.

ing computationally efficient. While the general structure of the algorithm (MSER computation, CNN classification, text line grouping) is quite standard, we have introduced a number of carefully designed improvements that have a significant effect in performance and speed. The novel contribution of this work includes a new strategy for thinning out MSERs that substantially reduces the cost associated with CNN with only minor loss in performance; a method for mining positive samples from word-level labeling; and the use of binarized patches for CNN classification, which is shown to improve results substantially with respect to grey-valued patches. While this algorithm is not yet feasible for real-time implementation on a smartphone, we are currently exploring further possibilities for increased speed-up that maintain similar quality level.

References

[1] H. Chen, S. S. Tsai, G. Schroth, D. M. Chen, R. Grzeszczuk, and B. Girod. Robust text detection in natural images with edge-enhanced maximally stable extremal regions. In *18th IEEE International Conference on Image Processing (ICIP)*, pages 2609–2612. IEEE, 2011.

[2] X. Chen and A. L. Yuille. Detecting and reading text in natural scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages II–366. IEEE, 2004.

[3] 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 *International Conference on Document Analysis and Recognition (ICDAR)*, pages 440–445. IEEE, 2011.

[4] P. Dollár, R. Appel, S. Belongie, and P. Perona. Fast feature pyramids for object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(8):1532–1545, 2014.

[5] B. Epshtein, E. Ofek, and Y. Wexler. Detecting text in natural scenes with stroke width transform. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2963–2970. IEEE, 2010.

[6] S. M. Hanif and L. Prevost. Text detection and localization in complex scene images using constrained adaboost algorithm. In *10th International Conference on Document Analysis and Recognition*, pages 1–5. IEEE, 2009.

[7] W. Huang, Z. Lin, J. Yang, and J. Wang. Text localization in natural images using stroke feature transform and text covariance descriptors. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1241–1248. IEEE, 2013.

[8] W. Huang, Y. Qiao, and X. Tang. Robust scene text detection with convolution neural network induced mser trees. In *ECCV 2014*, pages 497–511. Springer, 2014.

[9] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman. Reading text in the wild with convolutional neural networks. *International Journal of Computer Vision*, pages 1–20, 2014.

[10] M. Jaderberg, A. Vedaldi, and A. Zisserman. Deep features for text spotting. In *ECCV 2014*, pages 512–528. Springer, 2014.

[11] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the ACM International Conference on Multimedia*, pages 675–678. ACM, 2014.

[12] D. Karatzas, F. Shafait, S. Uchida, M. Iwamura, L. Gomez i Bigorda, S. Robles Mestre, J. Mas, D. Fernandez Mota, J. Almazan Almazan, and L.-P. de las Heras. Icdar 2013 robust reading competition. In *12th International Conference on Document Analysis and Recognition (ICDAR)*, pages 1484–1493. IEEE, 2013.

[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[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. Real-time scene text localization and recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3538–3545. IEEE, 2012.

[16] L. Neumann and J. Matas. On combining multiple segmentations in scene text recognition. In *12th International Conference on Document Analysis and Recognition (ICDAR)*, pages 523–527. IEEE, 2013.

[17] L. Neumann and J. Matas. Scene text localization and recognition with oriented stroke detection. In *IEEE International Conference on Computer Vision (ICCV)*, pages 97–104. IEEE, 2013.

[18] L. Neumann and J. Matas. Efficient scene text localization and recognition with local character refinement. *arXiv preprint arXiv:1504.03522*, 2015.

[19] D. Nistér and H. Stewénius. Linear time maximally stable extremal regions. In *ECCV 2008*, pages 183–196. Springer, 2008.

[20] A. Shahab, F. Shafait, and A. Dengel. Icdar 2011 robust reading competition challenge 2: Reading text in scene images. In *11th International Conference on Document Analysis and Recognition (ICDAR)*, pages 1491–1496. IEEE, 2011.

[21] K. Wang, B. Babenko, and S. Belongie. End-to-end scene text recognition. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1457–1464. IEEE, 2011.

[22] T. Wang, D. J. Wu, A. Coates, and A. Y. Ng. End-to-end text recognition with convolutional neural networks. In *21st International Conference on Pattern Recognition (ICPR)*, pages 3304–3308. IEEE, 2012.

[23] C. Wolf and J.-M. Jolion. Object count/area graphs for the evaluation of object detection and segmentation algorithms. *International Journal on Document Analysis and Recognition*, 8(4):280–296, 2006.

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

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

[26] A. Zamberletti, L. Noce, and I. Gallo. Text localization based on fast feature pyramids and multi-resolution maximally stable extremal regions. In *ACCV 2014 Workshops*, pages 91–105. Springer, 2014.

[27] Z. Zhang, W. Shen, C. Yao, and X. Bai. Symmetry-based text line detection in natural scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2558–2567, 2015.