UNIVERSITY OF CALIFORNIA
SANTA CRUZ

TEXT SPOTTING IN THE WILD

A dissertation submitted in partial satisfaction of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

COMPUTER ENGINEERING

by

Siyang Qin

June 2018

The Dissertation of Siyang Qin
is approved:

Table of Contents

List of Figures

List of Tables

Abstract

Text Spotting in the Wild

by

Siyang Qin

Detecting and segmenting text in natural images is a challenging task which may find application in multiple 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. In this thesis, I propose two efficient and effective text detection system, a new text stroke segmentation algorithm with state-of-the-art performance, and a novel encoder-decoder network architecture that can automatically remove text from image.

The first text detection algorithm is a region-based method, designed in a bottom-up manner. Characters are first detected before grouped into words. To find each character, Maximally Stable Extremal Regions (MSERs) is used to propose a large number of candidate regions which then feed to a Convolutional Neural Network (CNN) to filter out background regions. To improve the robustness and avoid the "tricky" post-processing (character grouping) step of the previous method, a cascaded fully convolutional networks (FCN) is proposed to predict the location of each word directly by utilizing the wide range of context information.

Segmenting text stroke from its background can benefits optical character recognition (OCR) and other tasks. I propose the use of FCN and fully connected

CRF with a novel pairwise kernel definition that includes stroke width information. In order to train the model, we create a new synthetic dataset with 100K text images. Our method outperforms the state-of-the-art algorithms while being more efficient.

Automatic removal of text or other objects from an image is considered an unsolved problem. It is challenging due to the fact that foreground segmentation is unknown, unlike the problem solved by traditional image inpainting algorithms which assume the known of where to reconstruct. In order to solve this challenging task, I introduce a novel encoder-decoder network architecture with two parallel and interconnected decoder branches, one designed to segment the foreground, the other to recover the missing background. The two decoders are connected via neglect nodes that determine which information from the encoder should be used for synthesis, and which should be neglected. The foreground text stroke segmentation and the synthesized background image are produced in a single forward pass.

To my parents, my wife Jia and my son Brandon for their endless love, support and company.

Acknowledgments

I would like to express my sincerest gratitude to my advisor, Dr. Roberto Manduchi, for his patient guidance and generous help over the past five years. His guidance and inspiring high-level vision helped me throughout my research. I owe him a big thank for his tremendous effort in helping me revise the papers and this thesis. Besides my advisor, I would like to thank Prof. Gabriel Elkaim and Prof. James Davis for serving on my dissertation reading committee and reviewing this thesis. Their enlightening comments are very helpful. I also want to thank Prof. Ricardo Sanfelice, Prof. Sri Kurniawan and Prof. Dimitris Achlioptas for serving on my advancement committee.

During my Ph.D. study, I am fortunate to collaborate with Seongdo Kim, Leo Neat, Ren Peng and Jiahui Wei from UCSC; Claude Knaus, Jason Chang, Radford Juang, Simon Safar and Wei Hong from Google. I also want to thank all the members from the Computer Vision Lab.

I would never be able to finish the degree without the support from my family. At the end, I would like to thank my parents, my wife and my son for their endless love, support and company.

Chapter 1

Introduction

1.1 Scene Text Detection

Fast automatic detection and reading of text (such a license plate number, a posted sign, or a street name) in images, is found useful for applications such as surveillance, forensics, autonomous vehicles, augmented reality (e.g., visual translation), and information access for blind people. Traditionally, OCR systems were designed for documents scanned into well-framed, good resolution images without excessive clutter, and taken under good illumination. For computational efficiency, a two-step process is often implemented. The first stage (text spotting) quickly detects the presence of areas in the image that are likely to contain text. These are then passed on to a recognition engine that decodes the textual content, using machine learning normally coupled with lexicon priors. The ability to detect individual words may simplify the work of the recognizer, and word-level detection is part of typical benchmarks such as the ICDAR incidental and focused datasets [1, 2]. Individual word detection could be cast as an object detection task, for example using popular algorithms such as as Faster R-CNN [68] or YOLO [66], that can directly predict the coordinates of each object using axis-aligned rectangular bounding boxes. Unfortunately, direct application of these algorithms to general text-bearing images produces unsatisfactory results [23, 78]. This is because general object detection methods have difficulties at detecting groups of very small objects such as words in a text line.

In this thesis, I present two methods on text spotting. The first one is a region based method (Chapter 2), its structure is shown in figure 2.1. The system uses multi-channel Maximally Stable Extremal Regions (MSERs) [49] to detect a large number of candidate character regions, then subsamples these regions using a clustering approach. Representatives of region clusters are binarized and then passed on to a neural 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. I also introduce a version that is three times as fast, with only a slight reduction in performance.

Region based strategy, however, was plagued by several drawbacks. Detecting individual characters is difficult in the presence of blur, noise, or poor contrast. Furthermore, region classification is performed using only local patch level information, text-like background elements were often confused with text characters without the use of wider range context information. Last but not least, in order to ensure good recall rate, many (possibly overlapping) templates must be processed by the CNN, resulting

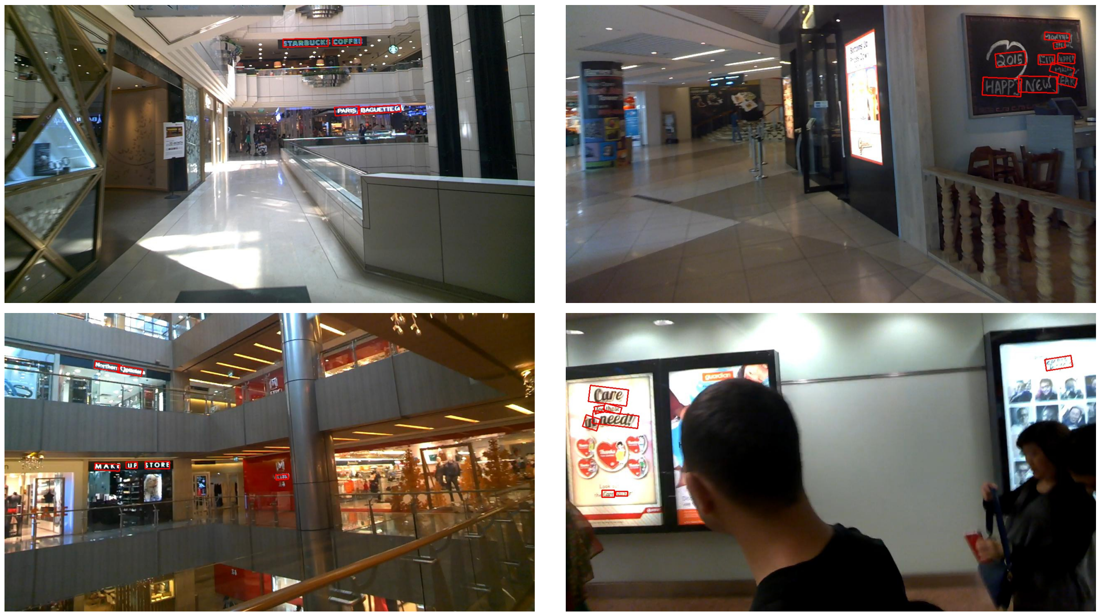


Figure 1.1: Our text detection results.

in long computational time.

To solve the above mentioned limitations, my second algorithm (Chapter 3) is formed by the cascade of two convolutional neural networks (see figure 3.1). The first segmentation network is fully convolutional and is in charge of detecting areas containing text (in pixel level). This results in a very reliable but possibly inaccurate segmentation of the input image. The second detection network analyzes each segment produced in the first stage, and predicts oriented rectangular regions containing individual words. No post-processing (e.g. text line grouping) is necessary. With execution time of 450 ms for a 1000×560 image on a Titan X GPU, our system achieves good performance on the challenging ICDAR incidental benchmarks [2].

1.2 Text Stroke Segmentation

Text stroke segmentation plays an important role in improving the performance of OCR [39] and other specific applications of interest. For example, binarization allows for operations such as text removal (and possibly substitution), text color change, and contrast enhancement. This type of operations are often required for stock photography processing (e.g., license plate number removal from Google StreetView images [19]), augmented reality (e.g., substitution of original text with its translation in a different language [18]), and assistive technology (e.g., to increase text readability for people with low vision [29]). Precise stroke segmentation is needed in these applications in order to preserve the naturalness of processed images.

Widely used techniques for text stroke segmentation include Maximally Stable Extremal Regions (MSERs) [49], a fast technique for generic local segmentation that is robust against domain and photometric distortions; and the Stroke Width Transform (SWT) [16], which is specifically designed for the detection of stroke like regions. Unfortunately, these methods give unsatisfactory results in challenging cases where low level image features become unreliable.

In this thesis, I propose a new technique for the accurate segmentation of text strokes from an image (Chapter 4). The algorithm takes in a cropped image containing a word. It first performs a coarse segmentation using a Fully Convolutional Network (FCN). While not accurate, this initial segmentation can usually identify most of the text stroke content even in difficult situations, with uneven lighting and non-uniform

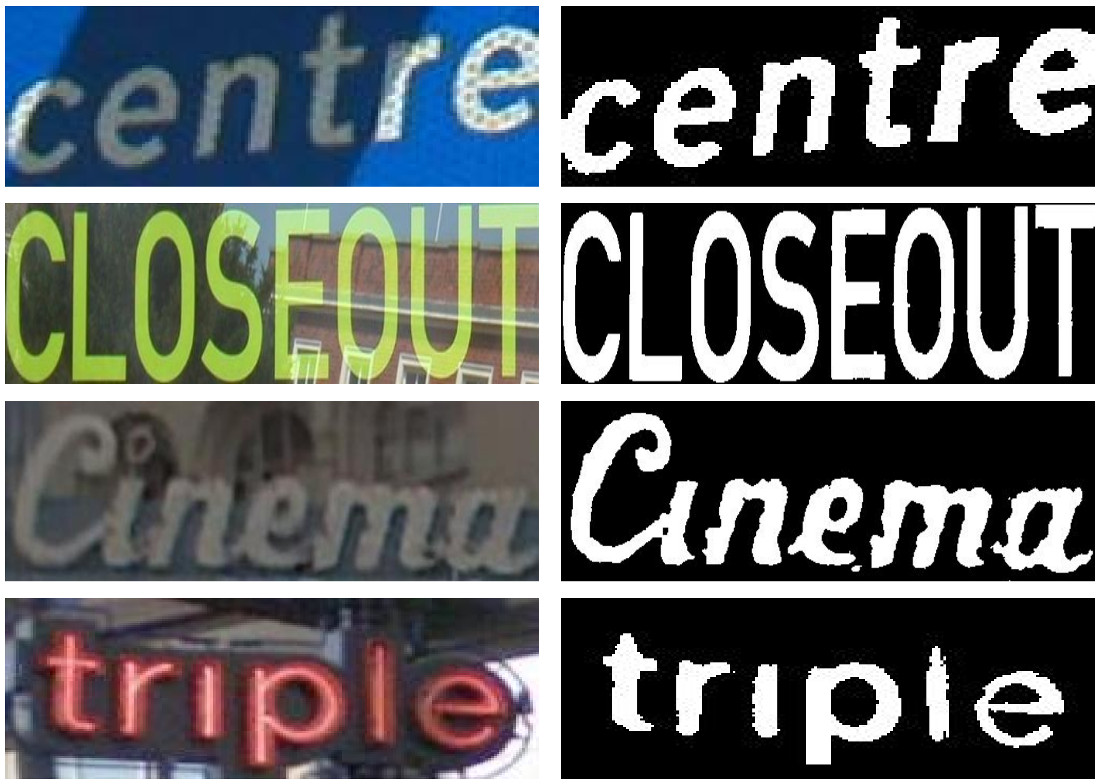


Figure 1.2: Text stroke segmentation is a challenging task due to large variance in text font, color, confounding background, poor contrast as well as different illumination conditions. Here we show several challenging cases and our results.

background. The segmentation is then refined using a fully connected Conditional Random Field (CRF) with a novel kernel definition that includes stroke width information. In order to train the network, we created a new synthetic data set with 100K text images. Tested against standard benchmarks with pixel-level annotation (ICDAR 2003, ICDAR 2011, and SVT) our algorithm outperforms the state-of-the-art algorithms by a noticeable margin.

1.3 Jointly Text Segmentation and Removal

Automatic removal of text content in an image is an extremely challenging task that can benefits lots of applications. There are many situations in which text needs to be erased, for example to protect personal information. Since automatic text removal is an yet unsolved problem, these operations are typically performed manually by skilled Photoshop editors or by simply blur the entire text area. Our interests in automatic text removal stems from the fact that it can be used to augment the existing limited and small scale scene text detection datasets by replacing existing texts on image with new texts (possibly in different languages).

Traditional image inpainting algorithms are designed to fill the missing region with realistic image content. Most of them require the known of the corrupted mask which indicating where to reconstruct, typically via manual input. In contrast, an automatic text removal system must be able to accomplish two tasks, text stroke segmentation and region filling. Although the two tasks can be performed in a cascade manner, our experiments show better result can be achieved by jointly predicting the foreground text segmentation mask and the recovered background image. My work is born from the realization that, for optimal results, these two tasks (segmentation and inpainting) should not be carried out independently.

We introduce a new system for automatic text removal and inpainting (Chapter 5). Unlike traditional inpainting algorithms, which require advance knowledge of the region to be filled in, our system automatically detects the area to be removed and

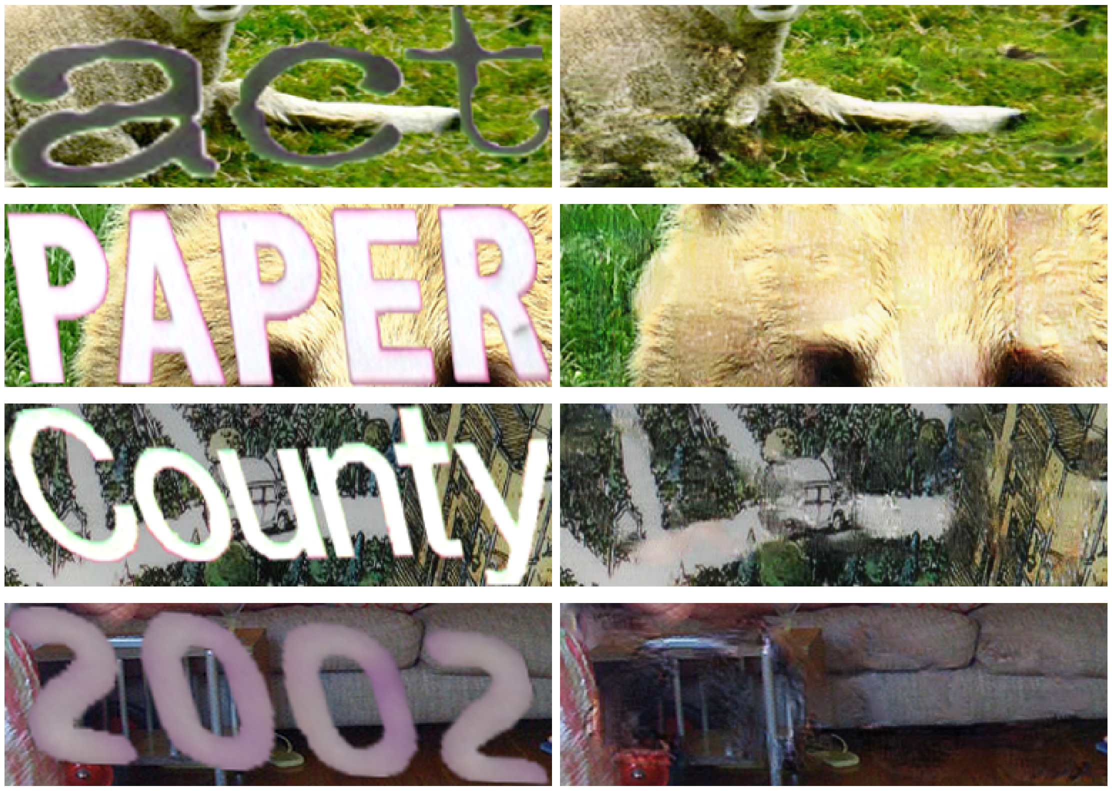


Figure 1.3: Our text removal results.

infilled. Region segmentation and inpainting are performed jointly in a single pass (see figure 5.1). In this way, potential segmentation errors are more naturally alleviated by the inpainting module. The system is implemented as an encoder-decoder architecture, with two decoder branches, one tasked with segmentation of the foreground region, the other with inpainting. The encoder and the two decoder branches are linked via neglect nodes, which guide the inpainting process in selecting which areas need reconstruction. In practice, neglect nodes provide *dec-fill* with information about which areas of the image should be erased and infilled, while preserving content elsewhere. The whole model is trained using a conditional GAN strategy. Comparative experiments show that our algorithm outperforms state-of-the-art inpainting techniques (which, unlike our system, do not segment the input image and thus must be aided by an external segmentation module.)

Chapter 2

Fast and Robust Text Spotting using MSER and CNN

2.1 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 [86]). 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 [99] 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. 2.1 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 MSERs [49], 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. 2.2). Our experimental results show that this simple "trick" can increase classification accuracy significantly. Rather than training the CNN 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.

2.2 Related Work

Text detection and localization methods have been traditionally grouped in two main categories. The first category contains sliding window approaches [83, 31, 99, 32, 8, 24, 82]. 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) [16] and maximally stable extremal regions (MSERs) [49, 57] 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. MSERs, 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 [54] proposed a method to perform efficient sequential selection from the set of extremal regions in the image to obtain character candidates. Huang *et al.* [27] 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.* [6] enhanced the MSERs 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 [53] proposed a novel approach to character detection based on the detection of strokes as connected components; strokes then induce a set of rectangles to be classified, thereby reducing the number of candidate character regions by three orders of magnitude with respect to traditional sliding-window approaches. Zamberletti *et al.* [97] proposed a hybrid system that generates a text confidence map using a sliding-window classifier based on fast feature pyramid [14], 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) [38] [28, 83, 31, 99] 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.* [83] 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 [9]). Huang *et al.* [28] use MSERs to find candidate regions that are then classified by a CNN similar to that of [83]. Zhang *et al.* [99] rely on the spa-

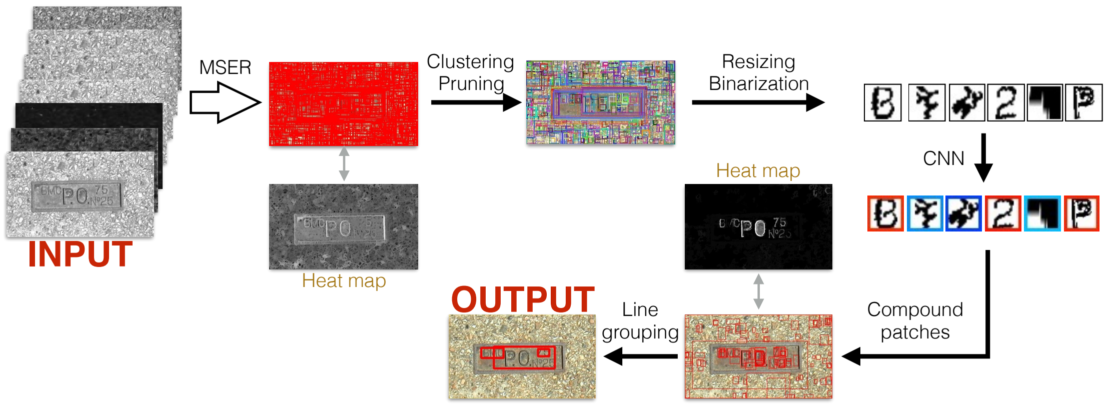


Figure 2.1: The overall structure of our first region based algorithm. Multichannel MSERs are thinned out through clustering and pruning, then resized and binarized before being fed to a CNN classifier. The compound patches that have been classified positively are then grouped into text lines.

tial symmetry that is charateristic of character groups, then use CNN to remove false positives.

For more comprehensive surveys, the reader is referred to [35, 72, 93].

2.3 Methodology

As customary for text spotting [8], we structure our algorithm as a cascaded classifier, with the initial stages designed to have high recall rate. Our system (Fig. 2.1) 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).

2.3.1 Stage I: MSER Segmentation

The first stage of our algorithm is multi-channel MSERs segmentation. MSERs [49], 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 MSERs computation: using the terminology of [49], 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 MSERs can be found that correctly encompasses just one full character. For this reason, several authors (e.g. [28]) 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 MSERs regions, and thus may contain digram or other n-grams. Consequently, we don't need to modify the regions produced by MSERs in any way (except for reshaping them to a common 32×32 size in Stage III.)

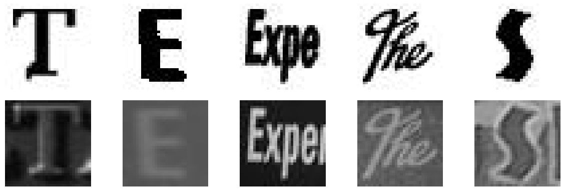


Figure 2.2: Some resized binary patches fed to CNN, together with their original grayscale counterpart. Note the presence of a trigram and of a four-gram.

2.3.2 Stage II: Region Clustering and Pruning

MSERs 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.

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.[54]). Measuring these features on all detected MSERs, however, wold 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. 2.1). In this context, a heat map simply assigns a value to each pixel equal to the number of MSERs that overlap on that pixel.

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

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 connected component, and specifically the one with highest fullness in the cluster. 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).

2.3.3 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 [83, 28] 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.

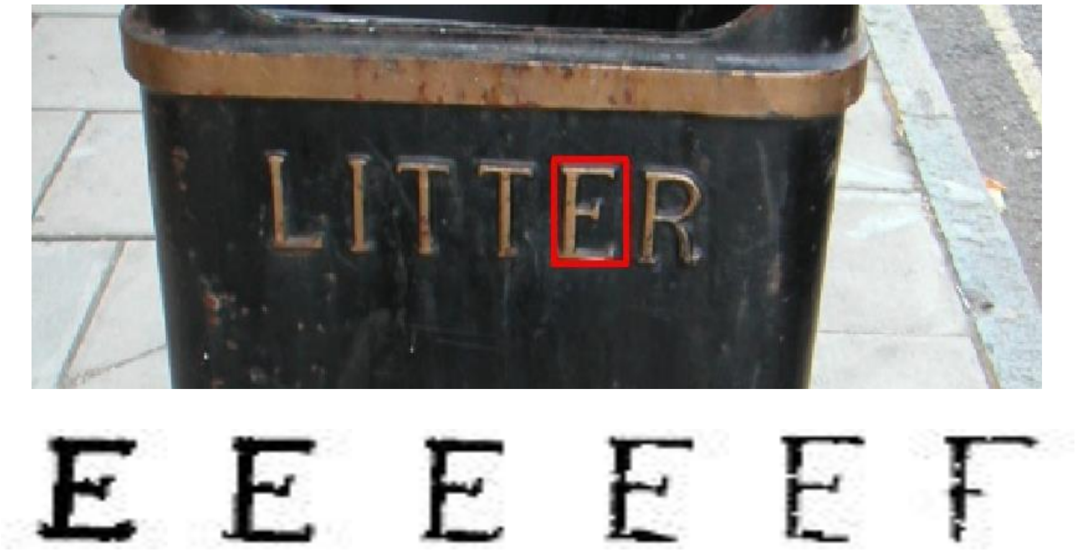


Figure 2.3: MSERs within a cluster centered at the red rectangle (only a few shown). The MSERs are ordered in decreasing value of *fullness*. The MSER to the left is chosen as the cluster representative

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. 2.4). 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 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. 2.2). We have found that this simply trick significantly improves results (see Sec.2.4). 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.

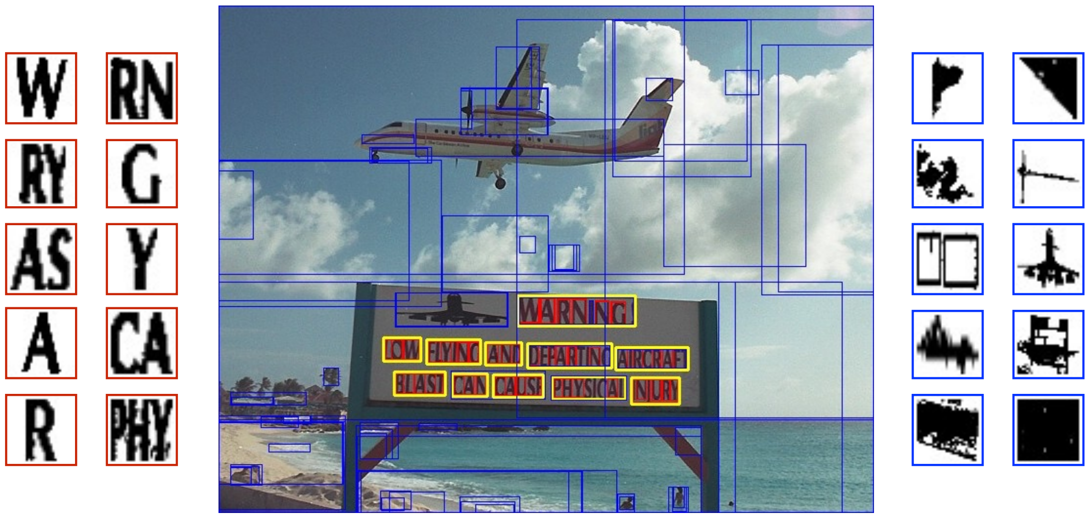


Figure 2.4: Mining positive (red) and negative (blue) examples from an image that was labeled at word level (yellow rectangles). Only a subset of MSERs are shown on the image for readability.

2.3.4 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.

2.4 Experimental Results

Our text spotter was implemented in C++ using OpenCV and the Caffe implementation of CNN [33]. 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.

2.4.1 Speed

End-to-end computational times for VGA image size are reported in Fig. 2.5 and Tabs. 2.1 and 2.2. 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 away (our "fast" implementation in Tabs. 2.1 and 2.2), only 300 patches are sent on to CNN, reducing its cost to 210 ms. Line grouping (Stage IV) takes 30 ms.

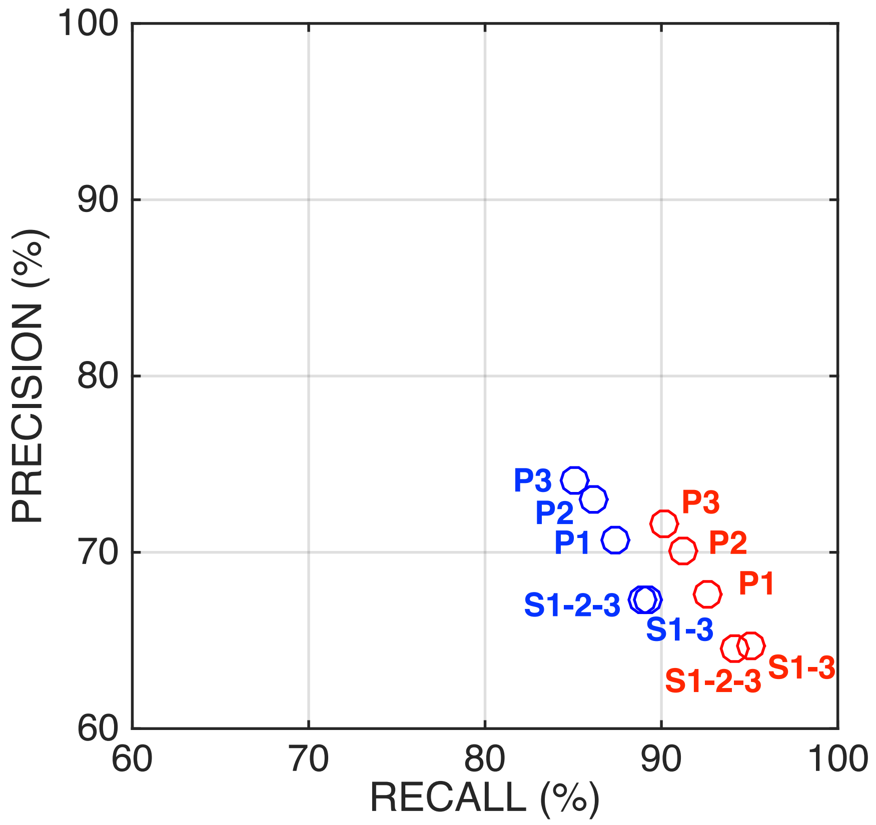


Figure 2.5: Patch-level evaluation (see Sec. 2.4.2 for definition of precision/recall in this context). Red: using binary patches. Blue: using greyscale patches. S1-3: Stages I and III with no MSER clustering (end-to-end computational time: 2.3 s/frame). S1-2-3: Stages I, II and III, with one representative 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).

2.4.2 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 [86] 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.* [99] appropriate for 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

[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.

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. 2.5 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. 2.3.2, 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 seen in recall 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.

2.4.3 Word-Level Evaluation

We benchmarked our system with the ICDAR 2011 and 2015 text localization data sets. Comparative results are shown in Tabs. 2.1 and 2.2; please note that the definition of precision and recall used for these tests [86] are substantially different from those introduced in the precious section. The slower version of our systems (without

	Recall (%)	Precision (%)	F-score (%)	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.* [99]	76	84	80	60*
Huang *et al.* [28]	71	88	78	unknown
Yin *et al.* [95]	68	86	76	0.43
Neumann *et al.* [52]	68	85	75	3.1

Table 2.1: Word-level benchmarking with the ICDAR 2011 text localization data set [86]. *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-score (%)	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.* [99]	74	88	80	60*
Zamberletti *et al.* [97]	70	86	77	0.75
Neumann *et al.* [55]	72	82	77	0.8
Yin *et al.* [95]	66	88	76	0.43

Table 2.2: Word-level benchmarking with the ICDAR 2015 text localization data set [86]. See caption of Tab. 2.1.

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. 2.6 shows several successful 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.

2.5 Conclusion

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

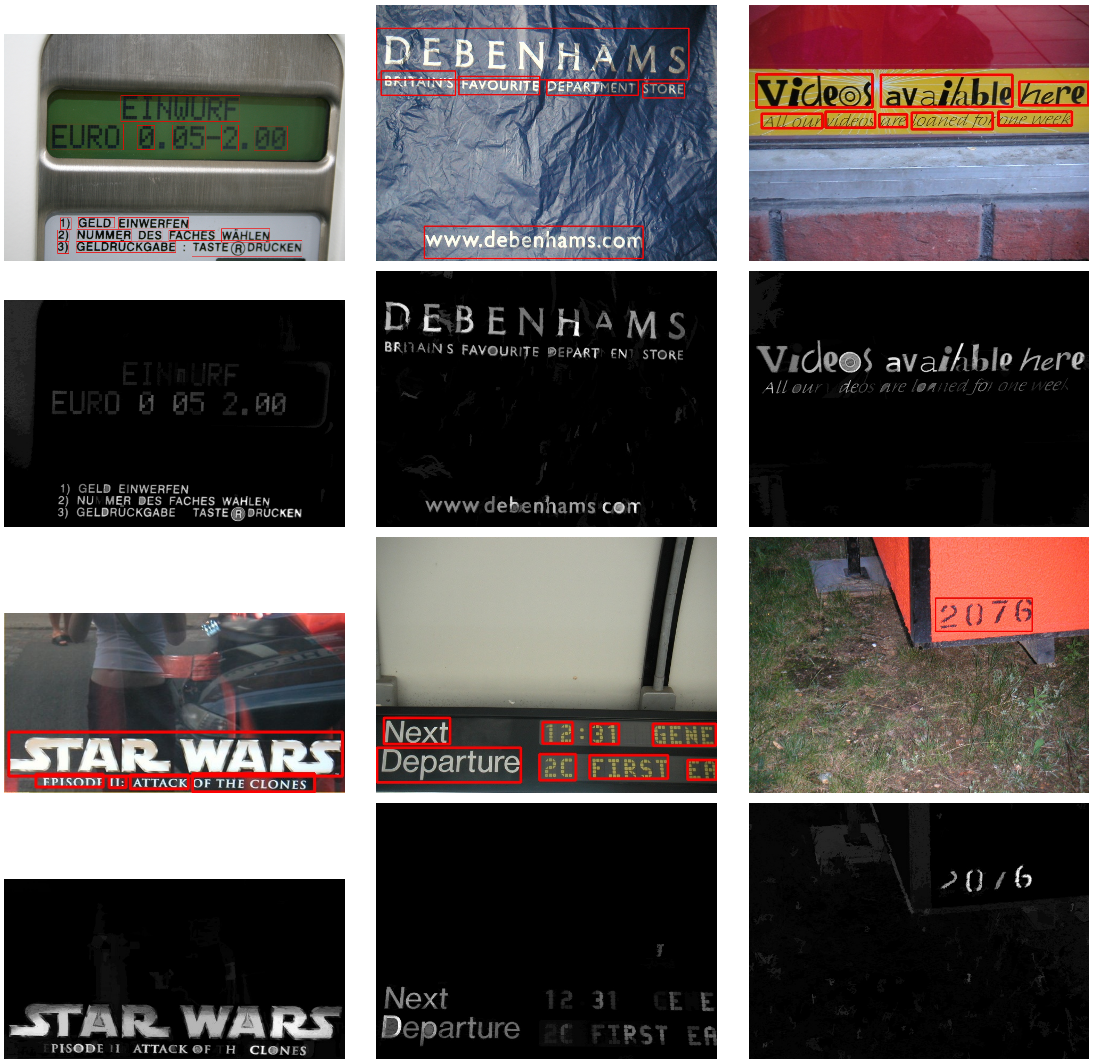

EINWURF
EURO 0.05-2.00
1) GELD EINWERFEN
2) NUMMER DES FACHES WÄHLEN
3) GELDRÜCKGABE : TASTE Ⓡ DRÜCKEN
DEBENHAMS
BRITAIN'S FAVOURITE DEPARTMENT STORE
www.debenhams.com
Videos available here
All our videos are loaned for one week
STAR WARS
EPISODE II: ATTACK OF THE CLONES
Next
Departure
12:31
GENE
2C
FIRST
2076

Figure 2.6: Some successful results from our algorithm. Top: image with superimposed detected word-level regions. Bottom: heat map.

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.

Chapter 3

Cascaded Segmentation-Detection Networks for Word-Level Text Spotting

3.1 Introduction

Fast automatic detection and reading of text (such a license plate number, a posted sign, or a street name) in images taken by a fixed or a moving camera, is very desirable for applications such as surveillance, forensics, autonomous vehicles [12, 13], augmented reality (e.g., visual translation), and information access for blind people. Traditionally, OCR systems were designed for documents scanned into well-framed, good resolution images without excessive clutter, and taken under good illumination. Recent mobile OCR software implemented in smartphones (e.g. ABBYY TestGrabber or KNFBReader), dedicated hardware (OrCam), or in the cloud (Google Vision API) produces very good results, but none of these systems is designed for real-time deployment (multiple frames per second), which is critical for the applications mentioned above. For computational efficiency, a two-step process is often implemented. The first stage (text spotting) quickly detects the presence of areas in the image that are likely to contain text. These are then passed on to a recognition engine that decodes the textual content, using machine learning normally coupled with lexicon priors.

This contribution focuses on fast and accurate word-level text spotting. The ability to detect individual words may simplify the work of the recognizer, and word-level detection is part of typical benchmarks such as the ICDAR incidental and focused datasets [1, 2]. Individual word detection could be cast as an object detection task, for example using popular algorithms such as as Faster R-CNN [68] or YOLO [66], that can directly predict the coordinates of each object using axis-aligned rectangular bounding boxes. Unfortunately, direct application of these algorithms to general text-bearing images produces unsatisfactory results [23, 78]. This is because general object detection methods have difficulties at detecting groups of very small objects such as words in a text line.

Another possible approach to text spotting is the use of segmentation algorithms (such as the fully convolutional networks, or FCN [44]) to identify images areas that are likely to contain text. These algorithms have proved very effective in terms of detecting text at variable size, but are generally poor at identifying individual words [100, 92].

In this work, we combine FCN's remarkable robustness at segmenting text regions, with YOLO's efficient mechanism for detecting objects (in this case, words), ap-

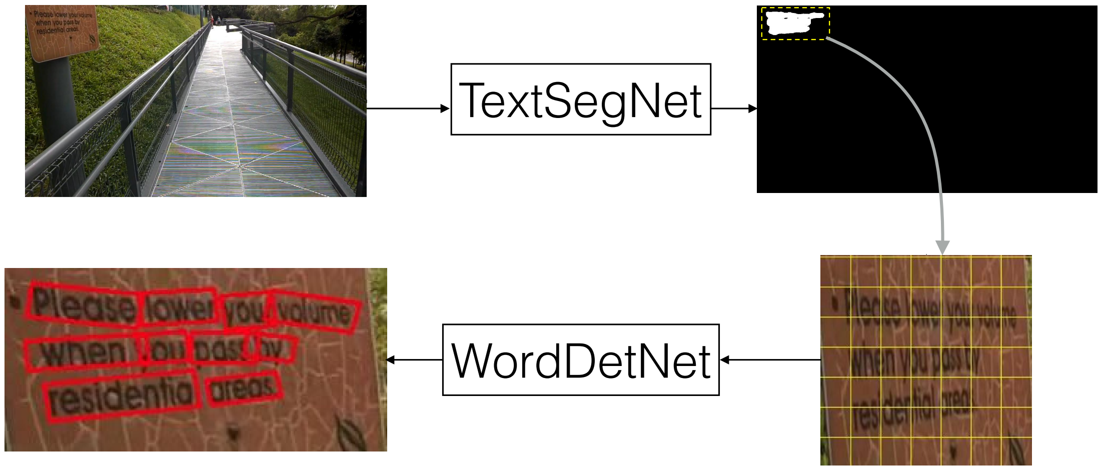


Figure 3.1: The general architecture of our word-level text spotting system. TextSegNet finds text blocks with arbitrary shapes and size. A squared resized block is passed on to WordDetNet, which generates oriented rectangular regions containing individual words.

propriately modeled as oriented rectangles. The two systems are integrated as a cascade (see Fig. 3.1): text regions produced by our fully convolutional network (TextSegNet) are cropped out of the image and resized to a square shape with fixed size. Then, a YOLO-like network (WordDetNet) is trained to generate oriented rectangular bounding boxes around each word. A simple non-maximum suppression stage takes care of overlapping boxes. In analogy with foveated vision, TextSegNet takes the role of a "spotter", determining regions of interest to be analyzed in detail by WordDetNet. The resized text regions contain a limited density of words, matching the inherently limited capacity of WordDetNet. The scheme is simple and elegant, and requires none of the post-processing steps (region grouping into straight lines, word splitting) that are typical of prior approaches. With execution time of 450 ms per image, our method achieves excellent results on popular benchmarks.

3.2 Related Work

Early attempts at text spotting used hand-designed features to capture characteristics of text images, both statistical (bimodal marginal brightness distribution) and morphological (uniform stroke width, connectivity, consistent width and height, alignment into text rows). Two of the most successful examples were [54], based on MSER segmentation, and [16], based on the stroke width transform. While these techniques worked reasonably well, a substantial increase in detection and localization accuracy was achieved with the use of convolutional neural networks (CNN). The first such methods [83, 28, 99, 62] used specific techniques to extract regions (*proposals*) with good likelihood of containing text (or individual characters), which were then passed on to a CNN classifier that would rule out false detections. This strategy, however, was plagued by several drawbacks. Detecting individual characters is difficult in the presence of blur, noise, or poor contrast. Since the operators used for character detection were typically local, text-like background elements were often confused with text characters. In order to ensure good recall rate, many (possibly overlapping) templates must be processed by the CNN, resulting in long computational time.

Recent progress in semantic segmentation and object detection has offered new tools that are well suited to text spotting. Fully convolutional networks (FCN) [44, 102, 7, 89, 61] and end-to-end object detection architectures such as Faster R-CNN [68] and YOLO [66] process a full image, and produce pixel-wise labelling or labelled regions containing objects of interest. In particular, through the use of skip layers, FCN are able to analyze an image using both large and narrow receptive fields, effectively encoding both local features and global context. Unfortunately, the segmentation produced by FCN, while highly reliable, cannot in general separate individual text lines or words, and further processing is required (see Fig. 3.2).

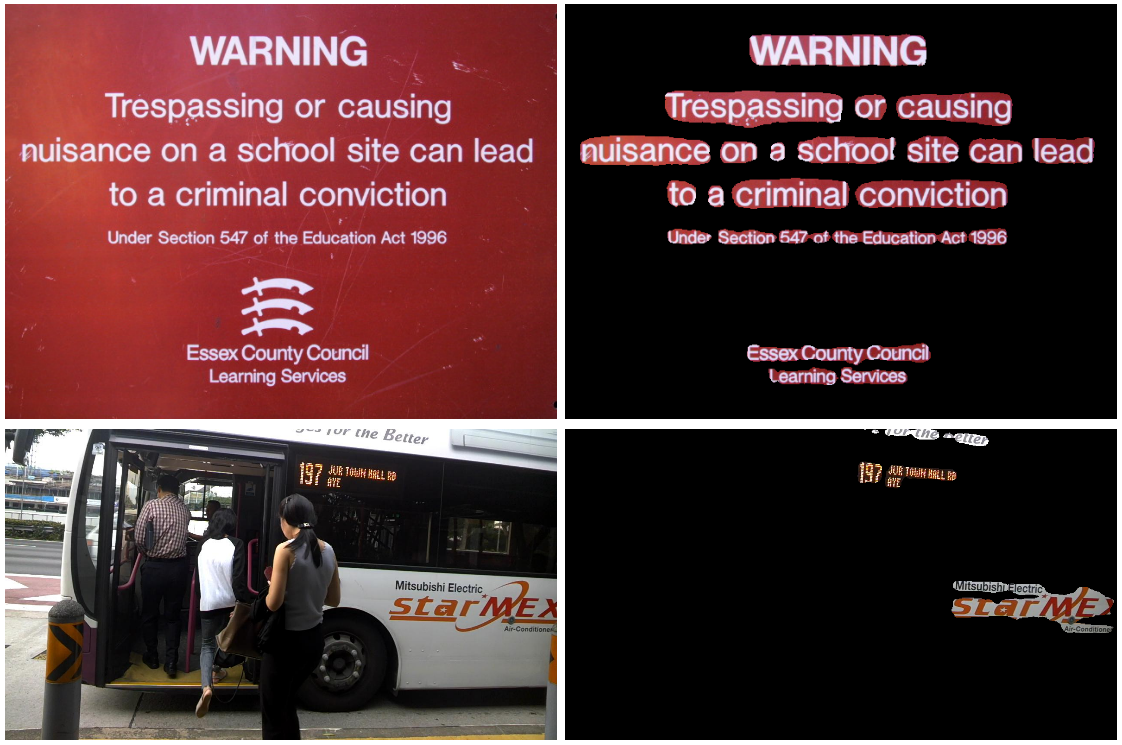
The use of FCN for text spotting was pioneered by Zhang *et al.* [100], who trained an FCN model to predict a saliency map; text line hypotheses were formed by combining this saliency map with individual character templates found via MSER. A final character-level FCN was used to remove false detections. The work of Yao *et al.* [92] added supervision on the scale and center of characters as well as the linking orientation of nearby characters when training the text block FCN. With additional information produced by FCN, the task of the text line grouping module was much simplified. Individual words were found based on the detection of indents between characters in a text line.

In order to accurately locate text lines, Tian *et al.* [78] used a recurrent neural network to connect proposal regions into individual lines. This algorithm could be trained end-to-end; compared to other bottom-up methods, it required no dedicated post processing to form a text line. Unfortunately, this method only worked for near horizontal text lines, and was unable to identify individual words. Gupta *et al.* [23] introduced a method for individual word detection that did not require any post-processing (such as grouping individual character templates). They trained a fully convolutional regression network similar to [66] using a large dataset of synthetic images. This algorithm splits an image into cells in a grid, where each cell is responsible for detection of a word centered in it. Each image cell predicts the pose parameters (location, width, height, and orientation) of a word, along with a confidence value. This method gives good results on relative simple benchmarks [1], but suffers from its inherent inability to detect small words in the image, when several such words locate inside the same (fixed size) image cell. This limits its performance on the challenging ICDAR incidental dataset [2].

3.3 Methodology

As discussed in Sec. 3.2, the traditional bottom-up approach (from individual characters to text line grouping to individual words) has recently been replaced by top-down strategies, which start by detecting text regions (e.g. using FCN), then proceed to identifying individual words. Unfortunately, while FCN enables very robust pixel-level text block segmentation, detecting individual text lines or words with the same mechanism becomes more challenging. This can be intuitively justified as follows.

FCN is designed to produce a pixel-level classification, where the label assigned to each pixel comes from a multi-scale analysis of the pixel's neighborhood (*scale* here identifies the effective size of each node's receptive field). The ability to utilize both local and extended context allows FCN to produce high quality semantic segmentation. Text patterns, however, are a special category of "objects", characterized by a specific geometric structure: characters are spaced regularly along a mostly straight line, with words in a line separated by relatively small gaps. With large receptive fields, it is hard for FCN to reliably separate individual words, resulting in segments that may contain a group of text lines, an individual line, or even an individual word or character (see Fig. 3.2).



Input image FCN segmentation

Figure 3.2: FCN is in general unable to separate individual words.

Direct application of object detection algorithms such as R-CNN or YOLO also generally produces poor results. R-CNN [20] relies on multiple region proposals, generated by methods such as selective search; this limits both performance and speed. Its successor, Faster R-CNN [68], replaces selective search with a learning-based algorithm for proposal generation. But due to the large variation in scale and aspect ratio that is typical of word regions, Faster R-CNN does not produce very good results, as reported in [78]. It should also be noted that most object detection algorithms predict axis-aligned rectangular regions, while words may have arbitrary orientation.

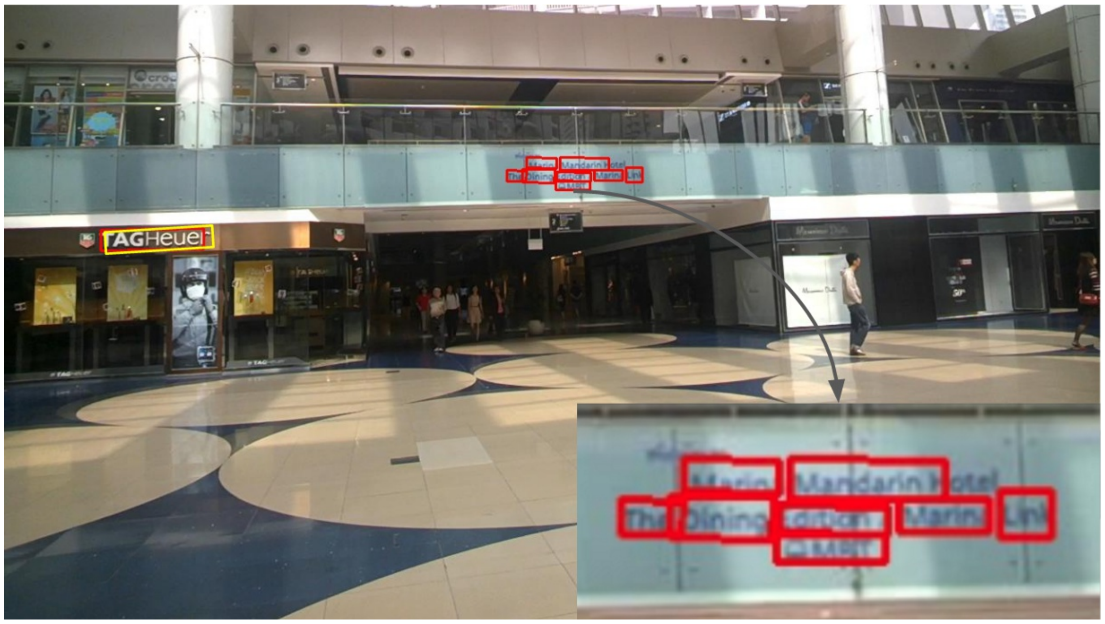


Figure 3.3: By cascading segmentation (TextSegNet) with detection (WordDetNet), our algorithm can detect both large and small words. If only WordDetNet is used, as trained on whole images, detection will fail in the case of too small words. Detected words by TextSegNet + WordDetNet are shown by red rectangles, the result of WordDetNet only are shown by yellow rectangles.

YOLO [66] formulates object detection as a regression problem. This algorithm is very fast and has shown good results for general object detection, but is by nature constrained in terms of its “capacity (the maximum number and density of regions produced in output). YOLO generates two boxes centered at each cell of a set defined on a regular grid. If more than two regions (e.g. words) are centered within the same cell, they cannot all be detected. The network capacity could be increased by changing the size of the cells or the number of boxes generated per cell. This, however, would result in increased computation and false positive rate.

Our architecture is a cascade of two stages (see Fig. 3.1): a segmentation stage (TextSegNet), that detects areas with high likelihood to contain text; and a word detection stage (WordDetNet), that, from the area cropped out by TextSegNet, resized to a common size, identifies and localizes individual words. The two stages are described in detail in the following.

3.3.1 TextSegNet

The first stage in our cascade, TextSegNet, is a fully convolutional network that takes both local and global context information into consideration to determine the label of each pixel. Text block detection is cast as a semantic segmentation problem with two labels ('text' and 'non-text').

3.3.1.1 Architecture

TextSegNet is based on the FCN-8s [44] model, which is derived from the VGG 16-layer network [73] with the final classifier layer removed, and the fully connected layers converted to convolution layers. We attach an additional final 1×1 convolution layer with channel dimension 2 to obtain prediction scores for 'text' and 'non-text'. Two skip layers are used to combine finer details (*pool 3* and *pool 4*) with high level semantic information; the output of the main and skip layers are combined and interpolated to the original image resolution using bilinear kernels (the weights of these kernels are also learnt during training). Softmax loss is used for training. The output of the network is a binary map representing likely areas containing text. For more details about the network structure of FCN-8s, the reader is referred to [44].

3.3.1.2 Training

During training and testing, all input images are resized so that their largest side has length of 1000 pixels. Both datasets considered in the experiments have ground-truth word-level labelling. Specifically, individual words are labelled by axis-aligned rectangular bounding boxes in the ICDAR 2013 focused dataset, and by generic quadrilaterals in the ICDAR 2015 incidental dataset. In order to train TextSegNet, a binary mask is first created, where all pixels in the word labelled regions are marked as "text", while the other pixels are marked as "non-text".

3.3.2 WordDetNet

Each individual region identified by TextSegNet is first resized to a fixed square shape. More precisely, a square box tightly bounding and co-centered with the region is computed; the square box is then reshaped uniformly to a fixed size. In this way, the aspect ratio of the text image is not changed. The only exception is for segments that are very close to the image edges, where a co-centered square bounding box would extend outside the image area. In this case, a rectangular bounding box is considered, which is then re-sized as a square (see eg. Fig. 3.1). Reshaping text image segments to uniform size provides some degree of scale invariance, except for segments containing one or a few very long text lines, in which case the reshaped characters may have small size. The square reshaped images are then fed to a network (WordDetNet) to predict the locations of individual words.

Inspired by the YOLO architecture [66] and the work of Gupta *et al.* [23], WordDetNet is tasked with identifying individual words. It defines a $N \times N$ grid on the image, where each cell in the grid predicts B candidate oriented rectangular regions (boxes), all centered within the cell. A box can be parameterized in terms of the position (x, y) of its center relative to the bounds of the grid cell, its width and height (w, h) relative to the size of image, and its orientation angle θ, which is normalized to a value between 0 and 1. In addition, the network produces a value (C) that represents the confidence that this box actually contains a word.

3.3.2.1 Loss Function

The network is trained to minimize a multi-part squared loss function $L(\mathbf{P})$. $\mathbf{P}$ represents the set of all $N^2 \cdot B$ box parameter vectors $\mathbf{p}_i^j = (x_i^j, y_i^j, w_i^j, h_i^j, \theta_i^j, C_i^j)$, where i identifies the cell and j identifies the box. $L(\mathbf{P})$ is defined as follows:

$$\begin{aligned} L(\mathbf{P}) = \sum_{i=1}^{N\times N} \delta_i^{obj} \sum_{j=1}^{B} \Big[\delta_i^j L^{obj}(\mathbf{p}_i^j) + (1-\delta_i^j)(C_i^j)^2\Big] \\ +\lambda_{noobj} \sum_{i=1}^{N\times N} (1-\delta_i^{obj}) \sum_{j=1}^{B} (C_i^j)^2 \end{aligned} \quad (3.1)$$

$$L^{obj}(\mathbf{p}_i^j) = \left(1 - C_i^j\right)^2 + \lambda_{ang}\left(\hat{\theta}_i - \theta_i^j\right)^2 + \lambda_{coord} \cdot$$
$$\left[(\hat{x}_i - x_i^j)^2 + (\hat{y}_i - y_i^j)^2 + (\sqrt{\hat{w}_i} - \sqrt{w_i^j})^2 + (\sqrt{\hat{h}_i} - \sqrt{h_i^j})^2\right]$$

In the equation above, δ_i^{obj} is equal to 1 if the training image contains a word (represented by a oriented rectangle) centered at the i-th cell, 0 otherwise. δ_i^j is 1 only for the j-th predicted box with the largest Intersection-over-Union (IoU) with the word centered at the i-th cell (this is the *responsible* box [66]). The hatted notation represents "ground truth" values for the oriented rectangular word regions. In practice, for cells with no words centered on them, only the second term of the loss expression is activated, which penalizes predicted box confidence values larger than 0. Otherwise, only one responsible box is considered, with a penalty that takes into account both the box's confidence (which should be close to 1) and its localization accuracy; the non-responsible boxes are given a penalty for large confidence values. Note that Eq. (3.1) is equivalent to the loss function for the original YOLO network, except that (i) it contains an additional term for the rectangle orientation (θ), and (ii) there is no term assessing the posterior distribution of class assignment, as only one class (text) is considered here. We set the weights as follows: $\lambda_{ang} = 10$, $\lambda_{coord} = 5$ and $\lambda_{noobj} = 0.1$, which approximately balance multiple loss terms. The smaller value for λ_{noobj} is justified by the fact that only few cells are expected to be the center of words in the text region.

Predicted boxes with confidence value C_i^j less than 0.5 are discarded. Non-maximum suppression is used to remove overlapping detection with IoU less than 0.3. Note that general object detection algorithms use a larger threshold on the IoU. However, since words are normally placed along a line, a smaller overlap is expected in this case.

3.3.2.2 Architecture

WordDetNet (see Fig. 3.4) is built on the VGG 16-layer architecture [73], with the front layers initialized with the weights learned for TextSegNet (which also based on VGG 16-layer network). The original VGG 16-layer network is decapitated after *conv 5-3*, and two additional convolutional layers (*conv 6* and *prediction*) are added, along with a ReLU and dropout layer after *conv 6*. *conv 6* has 4096 filters with kernels size of 7×7, while *prediction* has $B \cdot 6$ filters with size of 3×3, resulting in B sets of box parameters ($\{\mathbf{p}_i^j\}$). Both *conv 6* and *prediction* are properly padded to maintain the size of the feature map. Note that that there are four 2×2 max pooling layers before *conv 5-3*. This means that, in order for the channels in output of *conv 5-3* to have size of $N \times N$ (corresponding to the cell grid described earlier), the input image must have size of $16 \cdot N \times 16 \cdot N$. For example, an input 240×240 color image will result in a grid of 15×15 cells. Note that the original YOLO network contains multiple fully connected layers, which are replaced by convolutional layers in WordDetNet. This results in a smaller number of parameters, allowing for easier training.

3.3.2.3 Training

WordDetNet operates on square text block regions. During training, text blocks are generated by the following algorithm, which mimics the expected output of a typical segmenter, where nearby words are likely to be grouped within the same segment. A graph is formed on the ground truth word-level rectangular bounding boxes, where two such bounding boxes are linked in the graph if their minimum distance is smaller

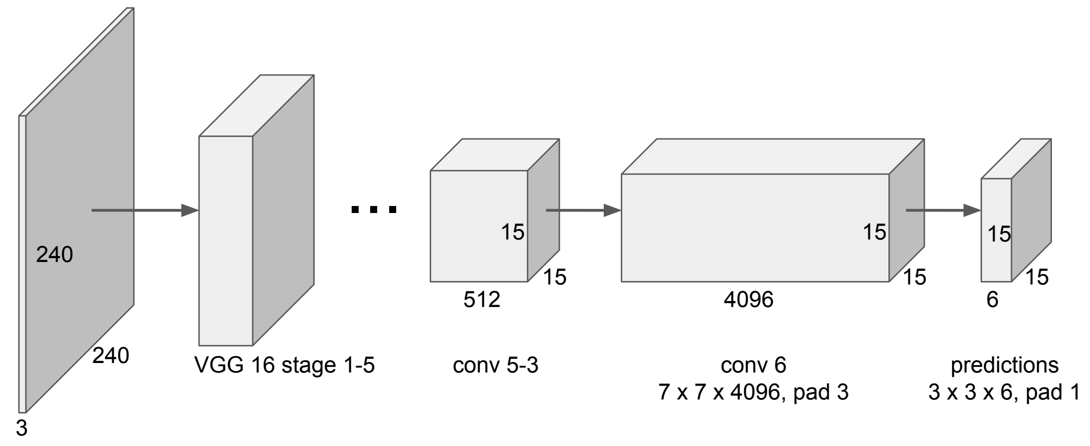


Figure 3.4: The structure of WordDetNet, shown here for N=15 and B=1. Our experiments show the above configuration achieves the best result.

than a threshold (set to be equal to the sum of the heights of the two rectangles). Then, the connected components of this graph are found. For each connected component, the tightest axis-aligned bounding rectangle is computed. Then, as explained in Sec. 3.3.2, this rectangle expanded to a square and resized, before being passed on to WordDetNet for training.

3.4 Experiments

3.4.1 Implementation Details

Our training data come from three sources: the training portion of the ICDAR 2013 focused dataset (229 images) and of the ICDAR 2015 incidental dataset (1000 images), as well as the large scale synthetic dataset (around 8 million images) described in [23]. We augment the images from the ICDAR datasets by means of random rotations, translations, and color adjustment, and add a subset of 20K images randomly selected from the synthetic dataset [23]. Note that we use only a small portion of the synthetic dataset in order to maintain a balance between natural and synthetic images. Overall, our training dataset contains about 35K images. We found that the addition of synthetic images has a moderate effect on performance (F-score increase by 1% in the ICDAR incidental dataset only).

The training dataset for WordDetNet contains 40K text blocks which are automatically mined using the algorithm mentioned earlier in Sec. 3.3.2.3. Weight sharing between TextSegNet and WordDetNet enables good performance in spite of relatively small training data size.

Our implementation is based on Caffe [33], and runs on a workstation (3.3Ghz 6-score CPU, 32G RAM, GTX Titan X GPU and Ubuntu 14.04). As mentioned earlier, the images given in input to TextSegNet are resized to 1000 pixels in their longer side. We use the same training strategy as in [44], with batch size of 1, learning rate of 10^{-9}, momentum of 0.99, and weight decay of 0.0005. Training TextSegNet takes 20 hours for 100K iterations. WordDetNet takes text blocks resized to 240×240 pixels, resulting in a 15×15 grid. Training parameters are: batch size of 16, learning rate of 10^{-5}, momentum of 0.9 and weight decay of 0.0005. At 50K iterations, training of WordDetNet takes 10 hours.

At deployment, one image is processed by TextSegNet in about 250 ms, then each text block is processed by WordDetNet in about 50 ms. End-to-end processing takes about 450 ms per image on average.

3.4.2 ICDAR 2015 Incidental Dataset

3.4.2.1 Dataset Description

The ICDAR 2015 incidental dataset [2] contains 1000 training images and 500 images used for testing. These images were taken by wearable cameras, without intentional focus on text regions. They are characterized by large variance in text size and orientation; some amount of blur is often visible. This dataset thus represents a much more challenging benchmark than the older ICDAR 2013 Focused dataset, which is described later in Sec. 3.4.3. A quadrilateral bounding box is defined for each word in the dataset; however, only the bounding boxes for the training portion are made available to the public. Detection results are evaluated by measuring the Intersection-over-Union (IoU) between a predicted box and the closest ground truth quadrilateral; if the IoU is larger than 0.5, the predicted box is deemed a true positive. A score is produced in terms of precision (ratio of true positives count and all detections count) and recall (ratio of true positives count and all ground truth labels count), as well as of their harmonic mean (F-score). Note that some unreadable words are marked as "do not care"; they are still counted as ground truth labels when computing precision, but not when computing recall. Our algorithm is only trained on the words not labelled as "do not care".

Table 3.1: Results on the ICDAR 2015 Incidental dataset

Method	Precision (%)	Recall (%)	F-score (%)
HUST [34]	44	38	41
AJON [34]	47	47	47
NJU-Text [34]	70	36	47
StradVision [34]	53	46	50
Zhang [100]	71	43	54
Google Vision API	68	53	59
CTPN [78]	74	52	61
Megvii-image++ [92]	72	58	64
Proposed (Seg+Det)	**79**	**65**	**71**
Proposed (Det only)	61	40	48

3.4.2.2 Results

Tab. 3.1 shows results of our method as compared with other state of the art published algorithms [100, 78, 92], as well as with Google Vision API, which produces word-level detection and recognition. Our cascaded (segmentation + detection) network outperforms the previous best results by a large margin (7% higher F-score than its closest competitor). In order to highlight the importance of the prior segmentation step, we also show results using only WordDetNet as applied on the whole image, rather than on the segments detected by TextSegNet. Performance decreases substantially in this case, showing the importance of a prior segmentation step. Some detection examples in challenging images are shown in Fig. 3.5.

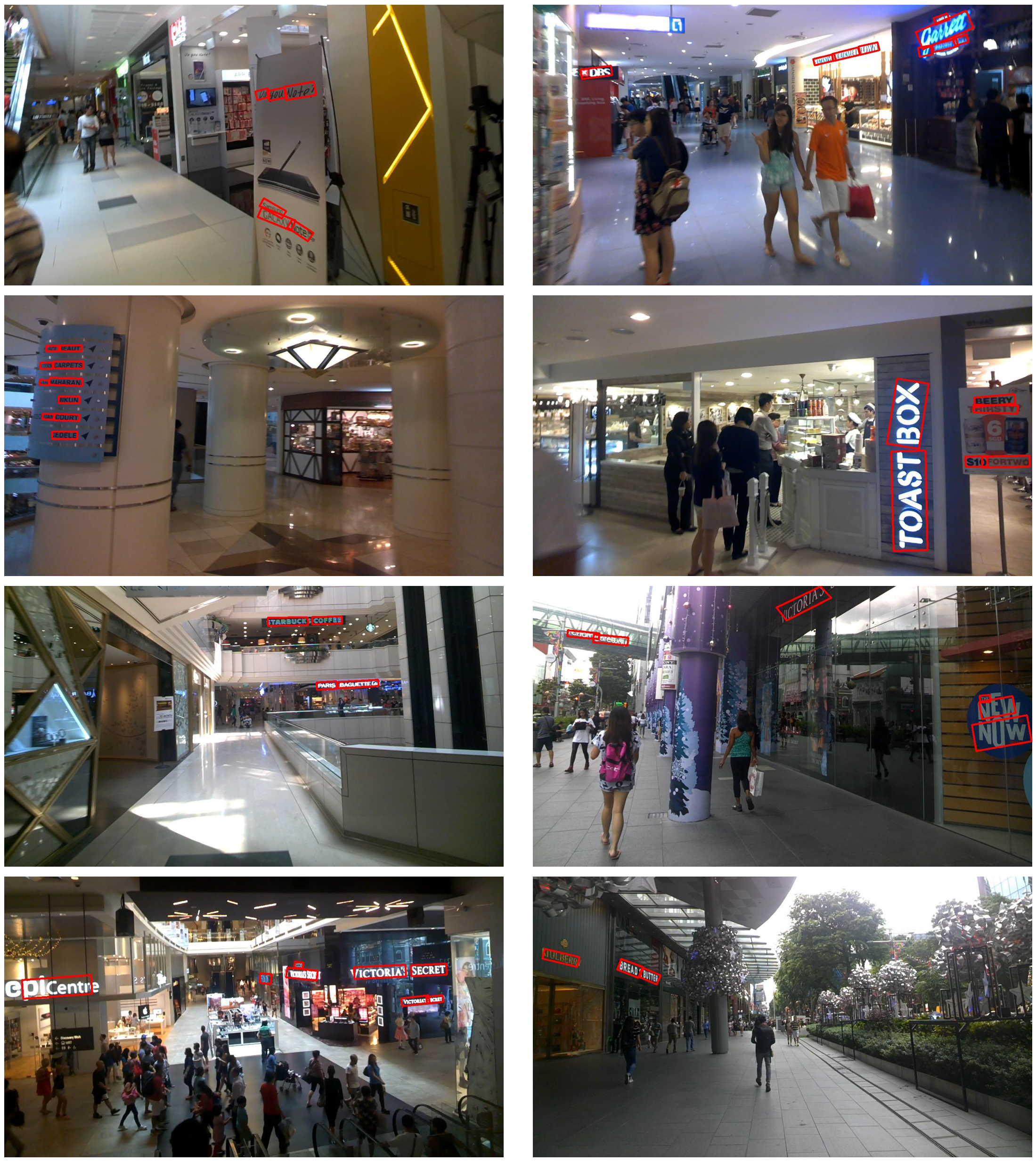


Figure 3.5: Some successful results.

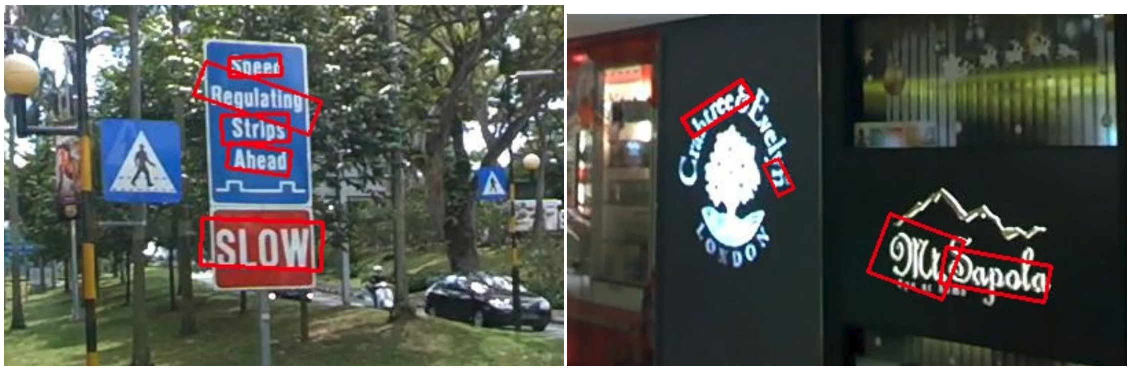


Figure 3.6: Examples of failure cases: incorrect box orientation or size, missing words in curved text.

The network capacities (defined as the maximum density of candidate boxes generated) can be changed by varying the number of cells in the $N \times N$ grid defined in WordDetNet, or the number B of boxes generated in each cell. Fig. 3.7 plots the F-score resulting from varying N between 7 and 19 (only odd values [66]) and with B equal to 1 and 2. This data shows that generating more than one candidate box per cell doesn't seem to provide an advantage, and that the optimal number of cells is 15×15. Note that with fewer cells, the burden is on the network to correctly localize the box within a larger cell. With more (hence smaller) cells, correct localization is easier, but the risk of false positives increases.

In spite of the good quantitative results, we noticed that sometime the box orientation and/or size as estimated by WordDetNet is somewhat inaccurate (see examples in Fig. 3.6). Incorrect orientation or size of an estimated word region may reduce the Intersection-over-Union with the corresponding ground truth region, thus affecting the resulting recall rate.

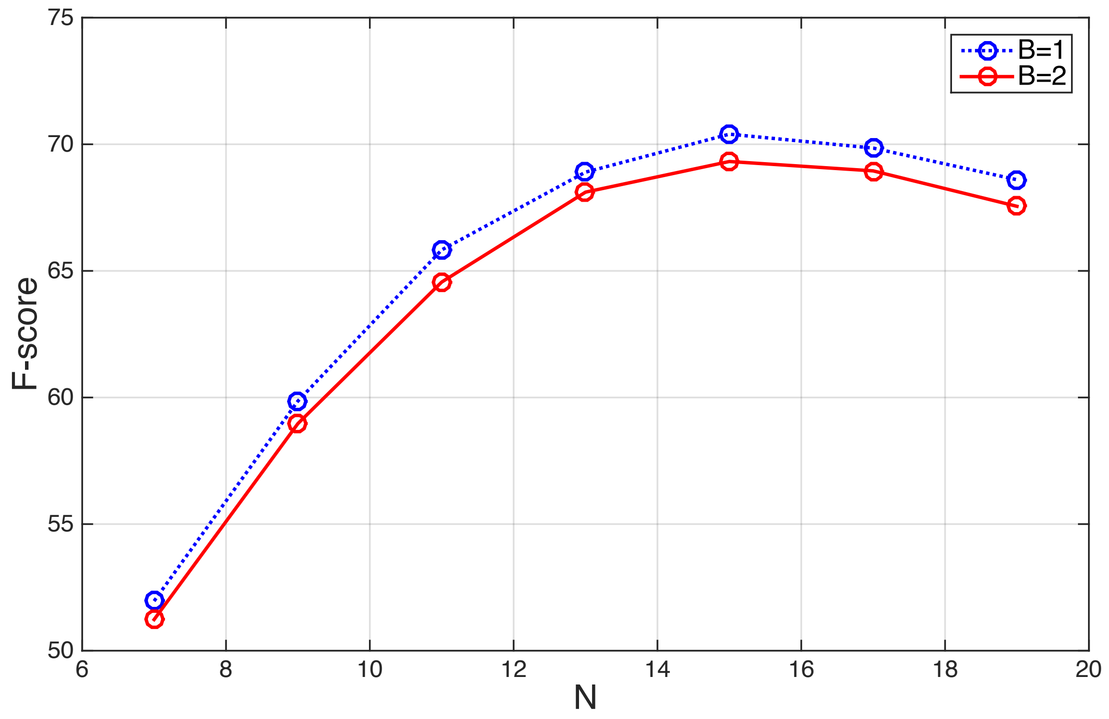


Figure 3.7: The F-score of ours system on the ICDAR 2015 incidental dataset as a function of the parameters N (which determines the number of cells in grid considered by WordDetNet) and B (the number of boxes generated per cell).

3.4.3 ICDAR 2013 Focused Dataset

3.4.3.1 Dataset Description

The ICDAR 2013 focused dataset [1] contains images “explicitly focused around the text content of interest”. 229 images are used for training and 233 for testing. In these images, text is seen at good resolution and at approximately horizontal orientation. Each word is labelled with an axis-aligned rectangle. The evaluation protocol is described in [35, 87].

Table 3.2: Results on the ICDAR 2013 Focused dataset

Method	Precision (%)	Recall (%)	F-score (%)
Neumann *et al.* [52]	85	68	75
Yin *et al.* [95]	86	68	76
FASText [5]	84	69	77
Huang *et al.* [28]	88	71	78
Zhang *et al.* [99]	88	74	80
TextFlow [77]	85	76	80
He *et al.* [25]	93	73	82
Qin *et al.* [62]	88	77	82
Zhang *et al.* [100]	88	78	83
Gupta *et al.* [23]	92	76	83
Yao *et al.* [92]	88	80	84
TextBoxes [40]	88	**83**	85
Zhu *et al.* [104]	**93**	81	87
CTPN [78]	**93**	**83**	**88**
Proposed	90	**83**	86

3.4.3.2 Results

Tab. 3.2 shows comparative results against other published algorithms. Our system has F-score of 86%, ranking among the top performers. The best current result is achieved by CTPN [78], with F-score of 88%. Note, however, that CTPN cannot identify individual words, and does not work for text with arbitrary orientation. In the more challenging ICDAR 2015 incidental dataset, our system outperforms CTPN by 10%. Looking closer at the data, one may notice that our algorithm produces the same recall value (83%) as the best performing systems, but lags behind in precision (90%, vs. 93% as obtained with CTPN [78] and Zhu *et al.* [104]). Part of the reason is that our method is able to find existing text that doesn't appear in the ground-truth labelling (see e.g. Fig. 3.8), resulting in an (incorrect) penalty in terms of precision.

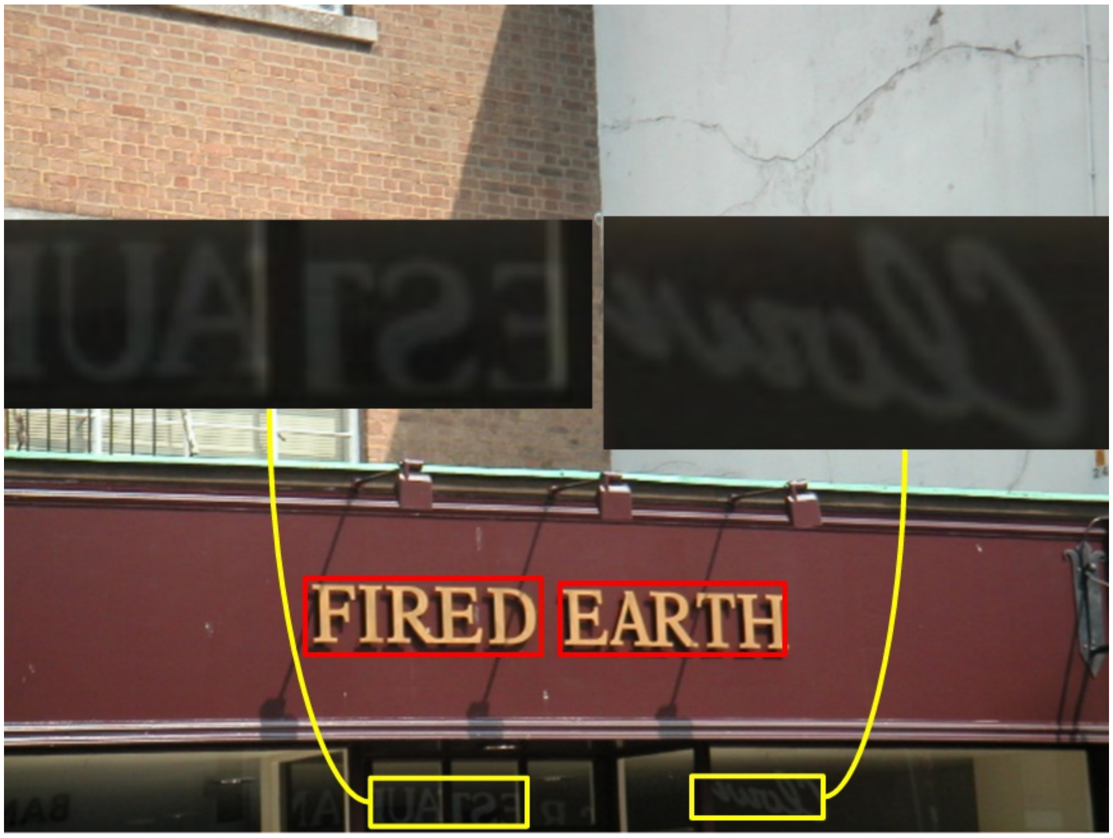


Figure 3.8: Our method finds words in very challenging situations (yellow boxes), even when these words are not labelled in the ICDAR 2013 focused dataset.

3.5 Conclusion

We have presented a new approach for word-level text detection and localization. Our algorithm identifies individual words and draws bounding boxes in the shape of oriented rectangles. The system is formed by the cascaded of a segmentation network (TextSegNet) and a detection network (WordDetNet), where the latter operates on regions segmented out by the former, resized to a common size. By combining segmentation and detection, we leverage on the strengths of each network. TextSegNet (which is based on the fully convolutional architecture of [44]) can very robustly detect the presence of text in the image, but is unable to identify individual words. WordDetNet (inspired by the YOLO architecture [66]) can effectively detect and localize individual words from a resized regions identified by TextSegNet. Unlike most existing algorithms, our system does not require a post-processing step to enforce alignment of the detected words.

We show experimentally that the first step (segmentation and resizing) is critical for the second step to be effective. On the challenging ICDAR 2015 incidental dataset, our system achieves top results among the published algorithms, outperforming the closest one by 7% in terms of F-score. In the more benign ICDAR 2013 focused dataset, our system produces an F-score that is only 2% less than the top performing algorithm (CTPN [78]).

Chapter 4

Robust and Accurate Text Stroke Segmentation

4.1 Introduction

Optical character recognition (OCR) has been one of the earliest success stories in computer vision. A fully electronic text reading system was demonstrated as early as in 1946 [48], while the first commercial OCR company, Intelligent Machines, was founded by Shepard and Cook in the early 50's. By the early 1980s, OCR of scanned documents was considered a solved problem; by the end of the same decade, a patent on automatic license plate reading was granted [21]. More recently, automatic text reading has received renewed interest in domains that were considered too challenging for traditional technology.

Scene text (or *text in the wild*) is a term often used to indicate text of any kind appearing in pictures or videos, often taken by hand or by a moving camera. As such, these images suffer from all sort of imperfections: blur, low resolution, poor exposure, reduced contrast. The text content itself is often very coincise (e.g., the name of a store), and not necessarily displayed on a straight line. Text may appear in front of a possibly multi-colored background. Specularities and cast shadows cutting across the text area are not unusual. Unlike scanned documents, which normally contain a large portion of well-structured text printed against a solid color background, detecting and localizing text areas in general scenes is challenging, especially when the scene contains visual clutter and the text itself occupies a small area. In addition, almost all applications involving scene text reading demand high frame rate processing. For this reason, considerable research effort went into algorithms for fast and robust scene text detection (or *spotting*). Once a text bearing region has been identified, its content can be processed by any standard OCR algorithm. Some text spotting algorithms specialize on separating individual words within the text area [63], thus further simplifying the job of subsequent modules.

Early attempts at text spotting considered an initial stage of *text stroke segmentation*, that is, segmentation of the regions corresponding to text strokes from the background. Widely used techniques include Maximally Stable Extremal Regions (MSERs) [49], a fast technique for generic local segmentation that is robust against domain and photometric distortions; and the Stroke Width Transform (SWT) [16], which is specifically designed for the detection of stroke-like regions. More modern text spotting approaches skip the text stroke segmentation step altogether, relying instead on general object detection techniques based on convolutional neural networks (CNNs).

While text stroke segmentation may not be needed for text detection, it still has an important role in improving the performance of OCR [39] and other specific applications of interest. For example, binarization allows for operations such as text removal (and possibly substitution), text color change, and contrast enhancement. This type of operations are often required for stock photography processing (e.g., license plate number removal from Google StreetView images [19]), augmented reality (e.g., substitution of original text with its translation in a different language [18]), and assistive technology (e.g., to increase text readability for people with low vision [29]). Precise stroke segmentation is needed in these applications in order to preserve the naturalness of processed images.

Our main contribution is a novel algorithm for text stroke segmentation that produces accurate results in the face of adversarial conditions such as cast shadows and cluttered background. The algorithm operates on image areas that have been previously identified as containing text (by an appropriate spotting algorithm). It is structured as the cascade of two modules. The first module uses a fully convolutional network (FCN) trained to robustly discriminate pixel within a stroke from those in the background. This results in a segmentation that reliably identifies text stroke areas; however, due to the multi-scale nature of FCN, this segmentation is often not accurate (see e.g. Fig.4.3). The second module is in charge of refining the earlier segmentation, in order to ensure that the contour of the stroke regions is correctly preserved. It relies on a fully connected conditional random field (CRF) model that uses an innovative expression for the pairwise energy term, one that uses information from the estimated local stroke width. We show that, by adding the proposed stroke width term to the more traditional bilateral kernel, the accuracy of segmentation improves noticeably.

In order to train the FCN, a large amount of images labeled at the pixel level are necessary. Unfortunately, existing data with pixel-level labeling of text content is scarce. We therefore assembled a new synthetic data set, with 100,000 images, representing a wide variety of font and backgrounds, along with pixel-level ground truth labels. This is the second original contribution of our work. Note that another synthetic data set was created in prior work to facilitate training of text spotters in natural images [23]. However, this prior data set did not provide pixel-level labels, and thus could not be used for our purpose.

We also propose the use of our text stroke segmentation algorithm and image inpainting technique to generate realistic synthetic data (see Sec.4.5.4). This is our third contribution.

4.2 Related Work

Document image binarization has a long history. A number of algorithms, based on image brightness thresholding , have been developed, beginning with Otsu' seminal work[59, 56, 71, 74, 26]. These techniques achieve good performance on scanned documents, but often fail on scene text segmentation, due in part to the typical large

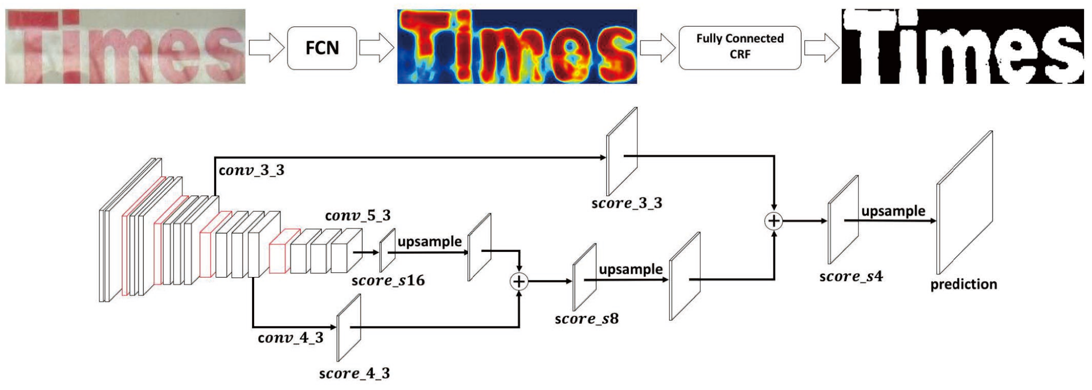


Figure 4.1: Our proposed framework.

variance in font, color, illumination that is typical in this type of imagery, as well as the to possible presence of complex background.

Early attempts at scene text segmentation tried to separate text strokes from background using local features such as edge [42] and color [84, 47], which were processed using simple thresholding or filtering. Later work used more sophisticated image models such as Markov Random Field (MRF) [51, 76, 50]. For example, Mishra *et al.* [51] used a MRF model where the unary energy term is described by a Gaussian mixture. The parameters of the color distribution within the text area were initialized using the stroke width transform (SWT [16]) Energy minimization was obtained via iterative graph cut [70]. A variant of this algorithm, proposed by Tian *et al.* [76], used the Stroke Feature Transform (SFT [27]) for initialization. SFT is more robust than SWT, resulting in more accurate initial color distributions, and thus avoiding the need for iterative graph cuts. Unfortunately, neither algorithm can cope with challenging situations, when local features become unreliable.

Maximally Stable Extremal Regions (MSERs) [49] have been used widely for scene text detection[54, 28, 62] and segmentation[54, 75, 52]. A classifier is trained to separate text from background based on the shape of each MSER region, along with other hand–drafted features. In order to achieve high recall rates, MSERs are often extracted from multiple color channels and using different thresholds; this, however, increases the computational load. Zhou *et al.* [103] proposed to use analysis-by-synthesis for text segmentation. A physical model was used to synthesize image given an initial set of rendering parameters and initial foreground/background labels. The parameters of the model were optimized using Expectation Maximization.

In recent years, convolutional neural networks (CNN) have been successfully applied to virtually all fields of computer vision, including scene text reading. In particular, fully convolutional networks (FCN) [44] are well suited for pixel-level segmentation. One problem with FCNs, however, is that, due to the large receptive field size of the cells in the network, the segmentation produced is often poorly localized (i.e., the contours of the detected regions may not closely follow the contours of the foreground regions in the image). Fully-connected conditional random field (CRF) [36] models are often used to overcome this limitation, and to refine local details of segmentation by minimizing a carefully designed global energy function. Chen *et al.* [7] fed the label assignment probabilities produced by an FCN to a fully-connected CRF, with the two modules trained separately. Zheng *et al.* [102] reformulated the mean-field algorithm for approximate inference for a fully-connected CRF as a Recurrent Neural Network (RNN), thus enabling end-to-end training.

4.3 A New Synthetic Text Data Set with Pixel-Level Labels

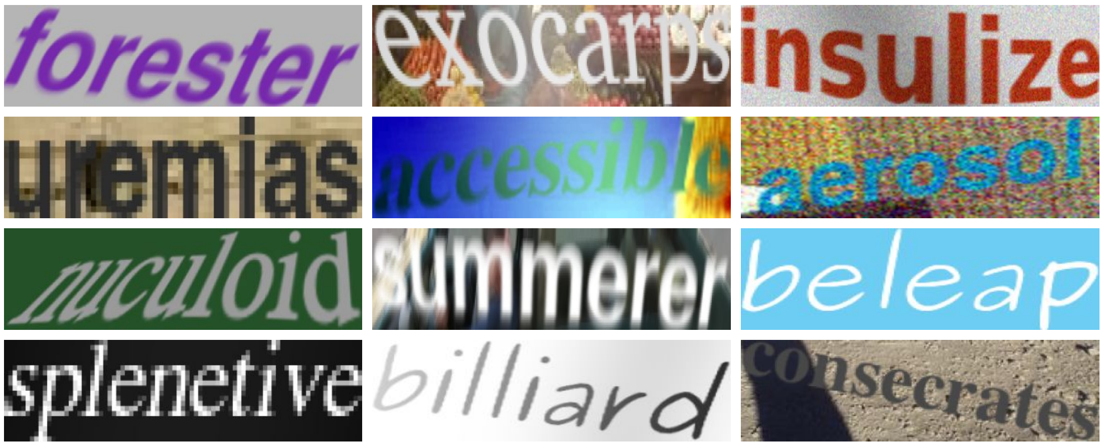


Figure 4.2: Samples from our synthetic dataset.

Training CNNs requires a large amount of data. Large data sets have been assembled for scene text detection (e.g. [81]). These sets are equipped with bounding box annotation identifying individual words. Unfortunately, data sets with *pixel-level* annotations are of a much smaller scale. For example, ICDAR 2003[46], ICDAR 2011[72], ICDAR 2015[1] and SVT[82] only contain a few hundreds word bounding boxes annotated at the pixel level as text stroke vs. background. While this size can be adequate for a test set, it is insufficient for training a network. We thus decided to generate a new, large scale data set with synthetic data. In the following, we describe how our new data set has been generated.

We began by sampling 100K words from an English corpus. These words were rendered using ImageMagick[1] onto a background. Each word was randomly assigned one of 264 different fonts, with height varying between 15 and 90 pixels. The font color could be white (25% of words), black (25%), grey (25%), or randomly chosen from a palette. Each word underwent one of a set possible geometric transformations (rotations, cylindrical projections, perspective transformations, wave distortion), with parameters sampled from a normal distribution. Words were then rendered against a background that could have a randomly chosen solid color (66% of words), or a portion of a "natural" image, randomly selected from the IAPR TC-12 Benchmark. The resulting images were corrupted with additive noise (Gaussian, impulse, and Laplacian), reflection and shadow effect. The resulting images have height of 112 pixels and variable width; the binary mask (text stroke vs. background) is provided for each image. We only use images from this set to train our algorithms (reserving a 10K subset for validation); the algorithms are then benchmarked on all available annotated real images.

4.4 Text Stroke Segmentation

Our proposed framework and FCN structure are shown in Fig.4.1. The resized input word patch (height is 112 pixels) is fed to the FCN to produce a coarse segmentation, more accurate text stroke mask is obtained with a fully-connected CRF refinement step.

[1]imagemagick.org

4.4.1 Coarse Segmentation: FCN

The first step in our algorithm is a coarse pixel-level segmentation of text strokes from the background using a FCN. Thanks to their ability to use information at multiple scales, FCNs can segment text strokes even in challenging situations.

The network structure of the original FCN [44] was derived from the VGG 16-layer network[73], with the final classifier layer removed, and the fully connected layers converted to convolutional layers. We modified the original FCN scheme for our application as follows. First of all, we remove the last pooling layer (*pool5*) and all subsequent layers. This is justified by the observation that text stroke segmentation from an already cropped word patch is a simpler undertaking than generic semantic segmentation, which was the task addressed by [44]. The last convolutional layer (*conv_5_3*) is fed to a 1×1 convolutional layer with channel dimension of two, producing class prediction scores for text and background (*score_s16*). As suggested in [44], two skip layers are added, with the purpose to combine low resolution, highly semantic information with finer detail. The coarse prediction scores *score_s16* are upsampled by two before being combined with the prediction scores from *conv_4_3* to produce a finer scale prediction (*score_s8*). The same process is repeated for the second skip layer. The resulting prediction score *score_s4* is then upsampled by four to match the input image size. The upsampling layers are initialized with a bilinear interpolation kernel, whose weights are then learned during training.

Another difference with respect to the original FCN [44] is that the skip layers branch out at the end of a “block” of layers between two pooling layers, rather than at the beginning (layers labelled in red in Fig. 4.1). Skip layers are used to maintain information at higher resolution. The end layer of a block has the same resolution as the beginning layer, but may contain semantically richer information, and thus may prove a better candidate for a skip layer branching point.

Note that features at the coarsest scale (*score_s16*) have receptive field size of 192-by-192 , which is substantially larger than the height of input word block (112 pixels).

4.4.2 Refinement: Fully-Connected CRF with Stroke Width Kernel

The first stage FCN is able to segment out text strokes under a variety of font, color, illumination and background. However, as observed in Fig.4.3, the resulting segments are often not accurately localized. This is likely due to the large receptive field size of the nodes in the network, and to the fact that the result is upsampled from a low resolution map.

In order to refine the segmentation produced by FCN, we add a fully-connected CRF as a post-processing step. This produces a very noticeable improvement. A further improvement is obtained by modifying the the standard bilateral kernel [7] used to compute joint energy terms. Specifically, we propose to include in this term the estimated stroke width as a new text-specific feature. This is born by the observation that the stroke width is approximately constant within a text character. The standard bilateral kernel, which discourages assigning different labels to nearby pixels that have similar colors, fails to properly characterize the appearance of characters with large local color variations (e.g. as due to a cast shadow); background pixels with similar color as text region might be wrongly predicted as text (see Fig.4.4). By adding a measure of stroke width consistency in the joint energy term, CRF is more likely to correctly segment out whole characters and filter out background region with confounding color.

We define the following CRF energy function:

$$E = \sum_{i} - \log P(x_i) + \sum_{ij} \theta_{ij}(x_i, x_j) \tag{4.1}$$

$$\theta_{ij}(x_i, x_j) = \mu(x_i, x_j) w \exp\left(-\frac{|p_i - p_j|^2}{2\theta_\alpha^2} - \frac{|I_i - I_j|^2}{2\theta_\beta^2} - \frac{|s_i - s_j|^2}{2\theta_\gamma^2}\right). \tag{4.2}$$

where x_i and x_j are labels for pixels i and j, located at position p_i and p_j, with colors I_i and I_j, and associated stroke widths s_i and s_j. (The computation of stroke width is described later in Sec. 4.4.2.1.) In the unary energy term, $P(x_i)$ is the probability of pixel i having label x_i; this is computed from the score returned by FCN. More specifically, $P(t_i) = 1 - P(b_i)$ is the probability that pixel i belongs to a text stroke. $\mu(x_i, x_j) = 1$ if $x_i \neq x_j$, zero otherwise. $\theta_{ij}(x_i, x_j)$ for $x_i \neq x_j$ is the cost of assigning different labels to pixels i and j, which depends on the distance between pixels, their difference in color, and their difference in associated stroke width. The hyper-parameter w controls the weight of the joint energy term, while θ_α, θ_β and θ_γ controls the scale of each feature.

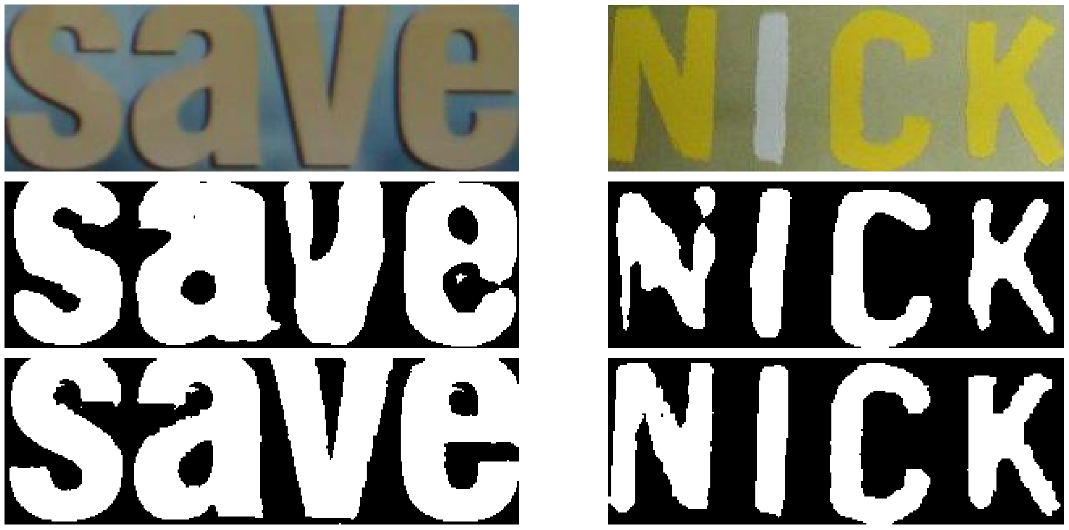


Figure 4.3: The first row is input images, second row is raw FCN predictions and last row contains final results with CRF refinement.

4.4.2.1 Modified Stroke Feature Transform

The stroke width at each pixel (term s_i in equation (4.2)) is computed before CRF refinement, based on the original image with additional input from the FCN predictions. It is based on the Stroke Feature Transform (SFT) [27], which is a modification of the original Stroke Width Transform (SWT) algorithm [16]. In the SFT algorithm, edges are first extracted from the image (using Canny); then, a line is drawn from each edge pixel in the direction of the image gradient. The line is stopped as soon as it hits another edge pixel, or when the color of the current pixel differs from the median color of the pixels in the current segment by a large margin. The segment is then accepted if the image gradients at its endpoints point in approximately opposite directions. In addition, after all segments have been drawn, any segment whose median gradient orientation or colors is significantly different from that of its neighbors is discarded. Finally,

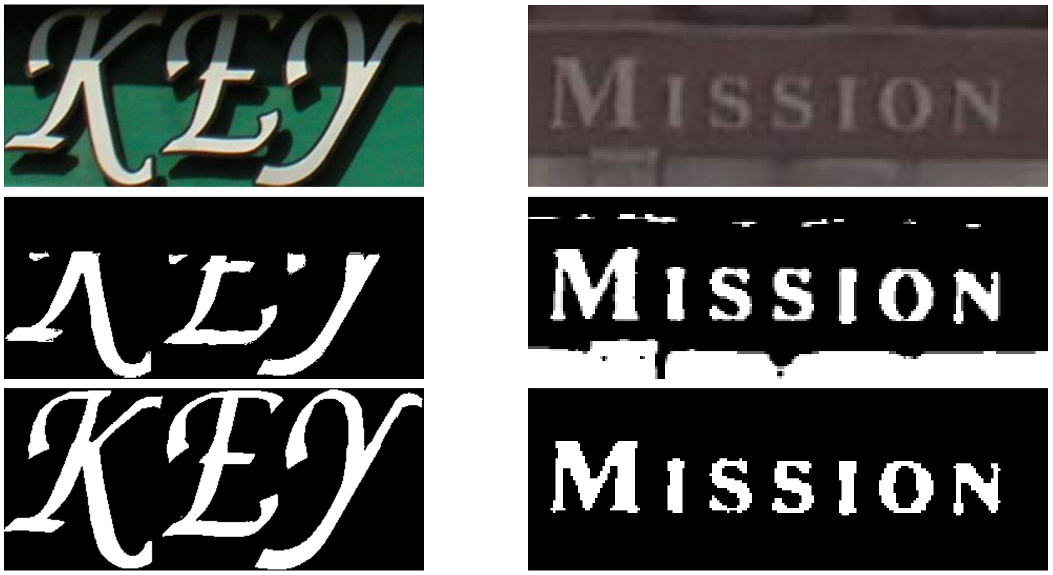


Figure 4.4: The first row is input images, second row is results without stroke width kernel and last row contains results with stroke width kernel. In left example the text color has large variance due to shadow and for right image the background contains regions with similar color as text region. In these cases use bilateral kernel alone becomes unreliable.

all pixels within a segment are assigned a *stroke width* value equal to the segment's length. Note that this algorithm may leave some small untouched "islands" of pixels within a character; these pixels are then assigned a value equal to the median of the value of their closest neighbors.

SFT was shown to be more robust than SWT, especially in situations in which edges may be difficult to compute reliably, or when the gradient at an edge pixel points away from the normal to the stroke edge. We further improve on the SFT algorithm (Modified SFT) by using information from the FCN output probability map. Specifically, we only keep a segment (as computed by SFT) when the average of $P(t_i)$ for pixels i within the segment is larger than a threshold. This helps ensuring that incorrect segments are not mistakenly accepted only because the image gradients at their endpoints happen to have approximately opposite directions. We run our modified SFT twice on two polarities, in order to find the stroke widths for dark text on light background as well for light text on dark background.

Note that, unlike SFT and SWT, we don't compute connected components of pixels with similar stroke width. We use the stroke width information solely as a feature in the kernel for the joint pairwise energy.

4.5 Experiments

4.5.1 Implementation Details

The coarse FCN segmentation component of our system is trained on the 100K synthetic images in our data set. As mentioned earlier, each image in the data set has fixed height (112 pixel) and variable width, depending on the word's aspect ratio. Due to the variable size of the samples, we set the batch size to one, and reshape the network at each forward pass. Cross-entropy loss is used during training.

The weights of our FCN are initialized from those of the network described in [63] (originally trained for scene text detection), and fine-tuned following the guidelines of [44]. We first fine-tune the model without skip layers for four epochs, with learning rate set to 10^{-9}, momentum set to 0.99, and weight decay set to 0.0005. We then add one skip layer at a time with reduced learning rate (10^{-11} and 10^{-12} respectively). The hyperparameters of the fully-connected CRF are determined by cross-validation on the validation set. We used the publicly available C++ implementation of the CRF's provided by the authors of [36].

Our system is implemented using Caffe [33] and runs on a workstation (3.3Ghz 6-core CPU, 32G RAM, Nvidia GTX Titan X GPU and Ubuntu 14.04 64-bit OS). At run tine, a 180 by 60 pixel input image is processed in about 0.2 seconds.

4.5.2 Quantitative Results

Data sets: We evaluated our algorithm against several popular document binarization methods [26, 56, 59], as well as against other state-of-the-art scene text segmentation techniques [45, 51, 17, 76, 75, 103]. We computed pixel-level precision, recall, and f-score for three popular scene text data sets: ICDAR 2003 [46] (1110 words); ICDAR 2011 [72] (716 words); and SVT [82] (647 words). For each data set, cropped rectangular regions containing individual word are available. Pixel level ground-truth labeling was generated by Kumar[39] using a publicly available semi-automated tool.

Polarity: For document binarization algorithms such as Niblack and Howe, correct polarity is not guaranteed. For fair comparison, we simply computed f-scores for each polarity, and reported the largest one. This is an optimistic measure: in practice, an automatic polarity check would be needed when using these algorithms, which may generate errors not considered by this measure.

Ablation study: We present results (1) using the full system (FCN+CRF/SFT), (2) removing the stroke width term form the CRF joint energy term (FCN+CRF), and (3) without using the fully connected CRF refinement step (FCN).

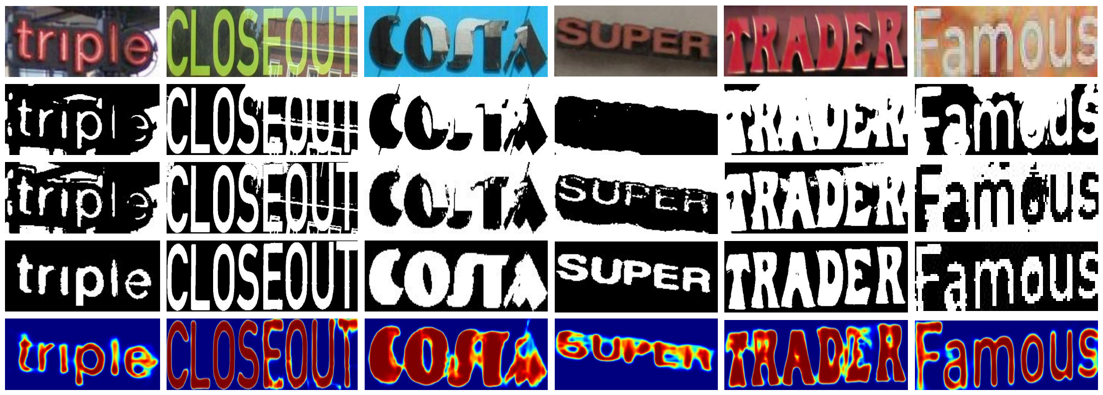


Figure 4.5: In this figure we compare the result of Otsu binarization algorithm [59], Zhou's text segmentation algorithm [103] and our method with several challenging images. From top to bottom: input image, Otsu result, Zhou's result, our result and our raw FCN output probability map.

4.5.2.1 ICDAR datasets

Comparative results for the ICDAR 2003 and ICDAR 2011 data sets are shown in Table 4.1 and 4.2. Note that these sets are relative easy as compared with SVT. Many images have clean background and clear text with bimodal color distribution, which allows simple binarization algorithm such as Niblack[56] and Howe[26] to reach appreciable performance (assuming correct polarity). More modern scene text segmentation algorithms consistently outperform these simpler binarization algorithms, thanks to their enhanced ability to remove background. FCN produces lower score than most competitors, due to poor localization. However, using the fully-connected CRF refinement step (FCN+CRF/SFT), significant improvement is observed, with an increase in f-score by 6.75% on ICDAR 2003 and by 7.05% on ICDAR 2011, achieving the the highest precision and f-score. Note that Lu[45] reaches a higher recall, but much lower precision due to oversegmentation. The stroke width term in the CRF kernel contributes to the improvement in f-score by 0.87% and 1.71% respectively. Detailed analysis shows that even though the modified Stroke Feature Transform sometimes fails with extremely low contrast images, the algorithm can still produce good results thanks to the robust FCN output and the bilateral CRF kernel component.

Table 4.1: Pixel level segmentation evaluation on ICDAR 2003 dataset.

Method	Precision	Recall	F-score
Niblack[56]	71.10	81.72	76.04
Lu[45]	72.61	**95.48**	82.49
Howe[26]	81.08	87.92	84.36
Mishra[51]	85.20	88.60	86.86
Feild[17]	86.58	87.84	87.21
Tian[76]	87.45	90.63	89.01
Zhou[103]	88.06	90.35	89.19
Tian[75]	88.27	90.18	89.21
Ours (FCN)	83.11	85.11	83.75
Ours (FCN+CRF)	88.80	90.47	89.63
Ours (FCN+CRF/SFT)	**89.96**	91.04	**90.50**

Table 4.2: Pixel level segmentation evaluation on ICDAR 2011 dataset.

Method	Precision	Recall	F-score
Niblack[56]	77.39	90.33	83.36
Lu[45]	77.26	**95.37**	85.36
Howe[26]	82.50	89.28	85.76
Feild[17]	90.84	89.61	90.22
Tian[76]	87.24	93.85	90.42
Ours (FCN)	84.87	86.91	85.88
Ours (FCN+CRF)	90.03	92.45	91.22
Ours (FCN+CRF/SFT)	**92.04**	93.84	**92.93**

4.5.2.2 SVT dataset

Compared with the ICDAR 2003 and 2011, the SVT dataset is arguably more challenging. Its images, which are extracted from Google Street View images, tend to be affected by noticeable blur, low contrast, complex background, and large variation in illumination. As shown in Table 4.3, prior methods produce results with substantially lower quality on this data set. Our system (FCN+CRF/SFT) achieves an f-score of 86.36%, compared with 81.20% for its closest competitor.

Table 4.3: Pixel level segmentation evaluation on SVT dataset.

Method	Precision	Recall	F-score
Niblack[56]	57.59	78.56	66.46
Howe[26]	69.16	81.32	74.74
Zhou[103]	71.93	87.18	78.82
Tian[75]	76.74	86.22	81.20
Ours (FCN)	77.67	82.43	79.98
Ours (FCN+CRF)	83.01	85.36	84.17
Ours (FCN+CRF/SFT)	**85.34**	**87.40**	**86.36**

4.5.3 Qualitative Results

In Fig. 4.5 we show some results of our method (FCN+CRF/SFT) for some challenging cases. The raw FCN probability map output is also shown, along with the output from the classic Otsu binarization algorithm[59] and from the scene text segmentation method proposed by Zhou [103]. Our method produce cleaner segmentation with high recall. It can deal with uncommon font, complex background, reflections, different illumination conditions and poor contrast. With large receptive field and trained multi-scale features, the coarse FCN segmentation produces robust results. The CRF refinement steps allows local details to be captured more faithfully. Key to our approach is the fact that "local" features (color, stroke width) are used only to refine the FCN output, and not to segment text from background, as in traditional algorithms based on MSER or edges. Even when local features become unreliable, FCN, thanks to its large receptive field, gets the job done.

Failure cases includes images with extremely low contrast (Fig. 4.6, left), as well as images with background similar to the text color and containing pattern consistent with text strokes (Fig. 4.6, right).

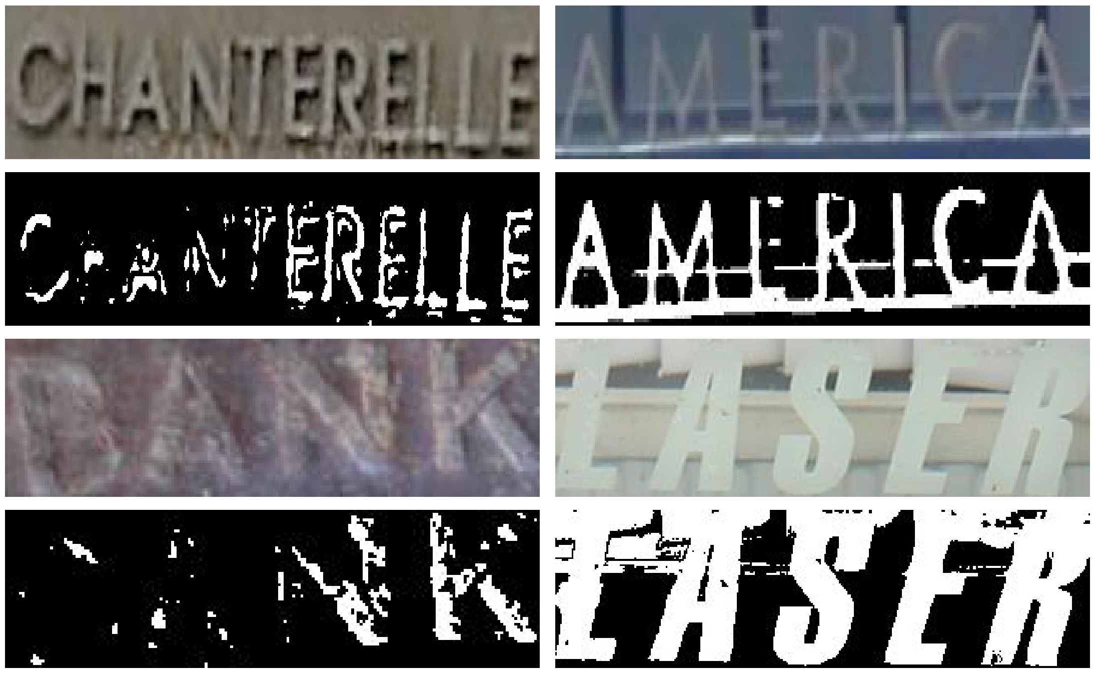


Figure 4.6: Failure cases.

4.5.4 Application: Text Substitution

Text substitution [18] is the art of replacing visible text in an image with other text (using different content, font, color, size, or language), in such a way that the rendered image looks “natural”. Accurate text stroke segmentation is an important component of text substitution. A typical computational pipeline for text substitution would follow these steps: (1) segment original text strokes; (2) remove text stroke content, substituting with background color or texture; (3) superimpose new text, possibly warped according to the surface orientation [23]. In Fig.4.7 we show examples of text substitution based on our text stroke segmentation algorithm. The segmentation was first morphologically dilated, then the resulting area was inpainted from nearby background using PatchMatch [3] (with 5×5 patch size).

One intriguing application of text substitution could be in the creation of natural-looking synthetic data sets for training convolutional neural network to perform text detection, segmentation and recognition. In [23], the author proposed a method to find “plain” surface on natural images to render text. However, the data set generated by [23] are not fully realistic, in that the distribution of background textures and context information on which text is superimposed may not match the distribution of real world scenario (see Fig.4.7 last column). By substituting text in regular scenes, we are able to generate new synthetic images (thus increasing the size of training data) while preserving the “natural” background. With our proposed method, large scale high quality synthetic dataset for multiple languages with character, word and pixel level ground truth labels can be generated.

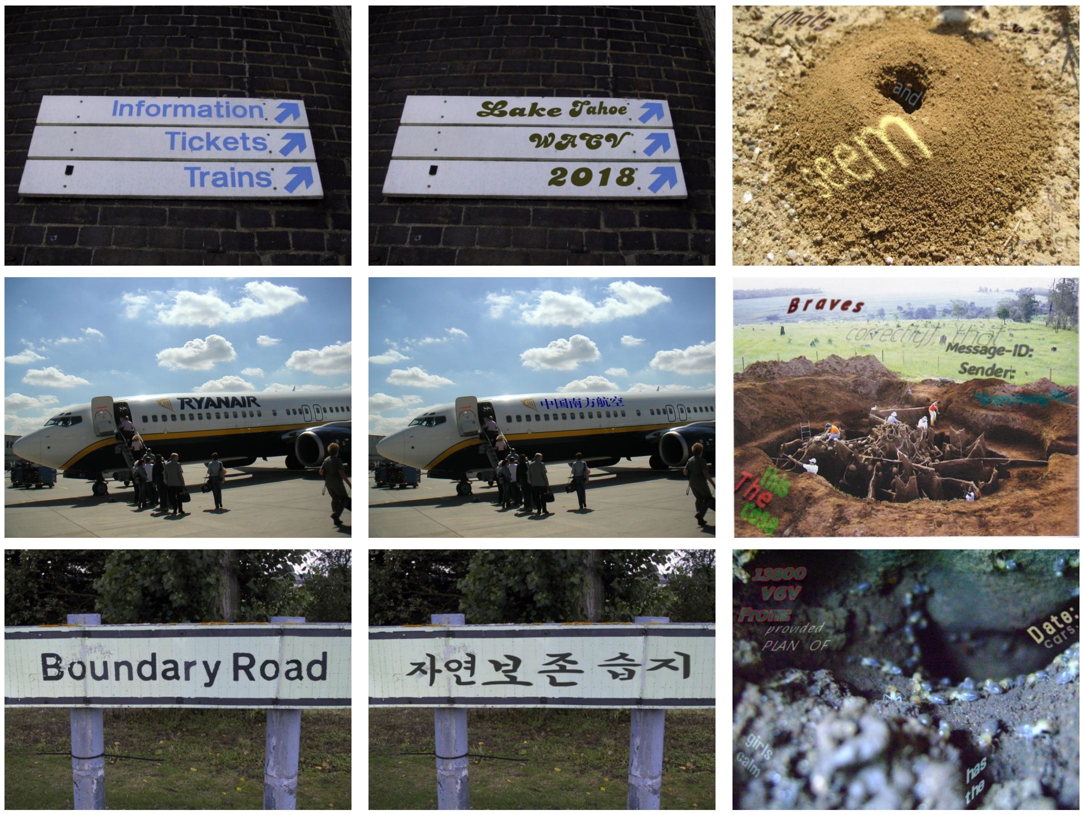


Figure 4.7: The left column is original images, middle column is our result with text substituted and last column contains samples from dataset generated in [23]. Clearly our synthetic data is more realistic.

4.6 Conclusion

We have presented a new algorithm for text stroke segmentation that produces state of the art results. The algorithms relies on FCN, a robust technique for pixel-level segmentation. FCN, however, cannot precisely localize the stroke edges, due to the large receptive fields of its cells and to its multi-resolution nature. The output of FCN is then refined by a fully connected CRF that uses the assignment probabilities from FCN as unary potentials. Results are further improved by adding to the standard joint energy term of the CRF information about the stroke width, which is computed using a modified Strike Feature Transform. The FCN is trained on a new data set with 100K synthetically generated test images. When tested on standard benchmarks with pixel-level annotations (ICAR 2003, ICDAR 2011, SVT), our algorithm is shown to work very well, with quality (as measured by the f-score) exceeding the state of the art by a sizable margin. We also show promising applications of our algorithm in text substitution.

Chapter 5

Automatic Semantic Content Removal by Learning to Neglect

5.1 Introduction

Automatic removal of specific content in an image is a task of practical interest, as well as of intellectual appeal. There are many situations in which a part of an image needs to be erased and replaced. This may include text (whether present in the scene or overimposed on the image), which may have to be removed, for example to protect personal information; people, such as undesired passersby in the scene; or other objects that, for any reasons, one may want to wipe off the picture. While these operations are typically performed manually by skilled Photoshop editors, substantial cost reduction could be achieved by automatizing the workflow. Automatic content removal, though, is difficult, and an as yet unsolved problem. Removing content and inpainting the image in a way that it looks “natural” entails the ability to capture, represent, and synthesize high-level (“semantic”) image content. This is particularly true of large image areas infilling, an operation that only recently has been accomplished with some success [60, 94, 91, 96].

Image inpainting algorithms described in the literature normally require that a binary “mask” indicating the location of the area to be synthesized be provided, typically via manual input. In contrast, an automatic content removal system must be able to accomplish two tasks. First, the pattern or object of interest must be segmented out, creating a binary mask; then, the image content within the mask must be synthesized. The work described in this paper is born from the realization that, for optimal results, these two tasks (segmentation and inpainting) should *not* be carried out independently. In fact, only in few specific situations can the portion of the image to be removed be represented by a binary mask. Edge smoothing effects are almost always present, either due to the camera’s point spreading function, or due to blending, if the pattern (e.g. text) is overimposed on the image. Although one could potentially recover an alpha mask for the foreground content to be removed, we believe that a more appropriate strategy is to *simultaneously* detect the foreground *and* synthesize the background image. By doing so, we do not need to resort to hand-made tricks, such as expanding the binary mask to account for inaccurate localization.

Our algorithm for content removal and inpainting relies on conditional generative adversarial networks (cGANs) [30], which have become the tool of choice for image synthesis. Our network architecture is based on an encoder-decoder scheme with

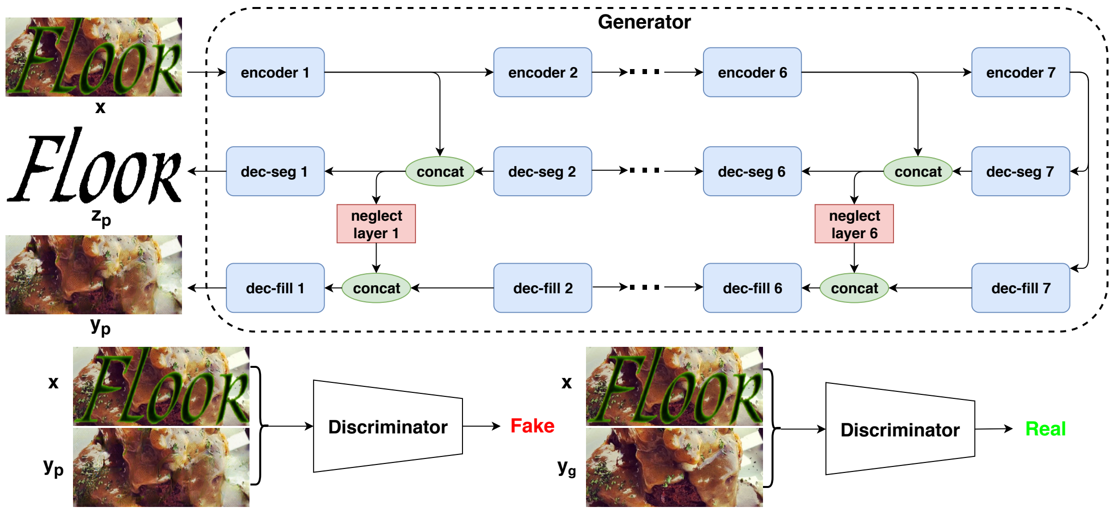


Figure 5.1: System architecture. x is the input image, y_p (recovered background image) and z_p (foreground text segmentation) are the outputs of the generator, y_g is the ground truth background image.

skip layers (see Fig. 5.1). Rather than a single decoder branch, however, our network contains two parallel and interconnected branches: one (*dec-seg*) designed to extract the foreground area to be removed; the other (*dec-fill*) tasked with synthesizing the missing background. The *dec-seg* branch interacts with the *dec-fill* branch via multiple *neglect nodes* (see Fig. 5.2). The concept of neglect nodes is germane (but in reverse) to that of mixing nodes normally found in attention networks [90, 80]. Mixing nodes highlight a portion of the image that needs to be attended to, or, in our case, neglected. Neglect nodes appear in all layers of the architecture; they ensure that the *dec-fill* branch is aware of which portions of the image are to be synthesized, without ever committing to a binary mask.

A remarkable feature of the proposed system is that the multiple components of the network (encoder and two decoder branches, along with the neglect nodes) are all trained at the same time. Training seeks to minimize a global cost function that includes a conditional GAN component, as well as L_1 distance components for both foreground segmentation and background image. This optimization requires ground–truth availability of *foreground* segmentation (the component to be removed), *background* images (the original image without the foreground), and *composite* images (foreground over background). By jointly optimizing the multiple network components (rather than, say, optimizing for the *dec-seg* independently on foreground segmentation, then using it to condition optimization of *dec-fill* via the neglect nodes), we are able to accurately reconstruct the background inpainted image. The algorithm also produces the foreground segmentation as a byproduct. We should emphasize that this foreground mask is *not* used by the *dec-fill* synthesizing layer, which only communicates with the *dec-seg* layer via the neglect nodes.

To summarize, this paper has two main contributions. First, we present the first (to the best of our knowledge) truly automatic semantic content removal system with promising results on realistic images. The proposed algorithm is able to recover high-quality background without any knowledge of the foreground segmentation mask. Unlike most previous GAN–based inpainting methods that assume a rectangular foreground region to be removed [60, 94, 91], our system produces good result with any foreground shape, even when it extends to the image boundary. Second, we introduce a novel encoder-decoder network structure with two parallel and interconnected branches (*dec-seg* and *dec-fill*), linked at multiple levels by mixing (neglect) nodes that determine which information from the encoder should be used for synthesis, and which should be neglected. Foreground region segmentation and background inpainting is produced in one single forward pass.

5.2 Related Work

Semantic image content removal comprises two different tasks: segmentation of the foreground region (which in some contexts represents a “corrupted” image region to be removed), and synthesis of the missing background after foreground removal. We briefly review the literature for these two operations in the following.

Pixel-level semantic image segmentation has received considerable attention over the past few years. Most recently published techniques are based on fully convolutional networks (FCN) [44], possibly combined with fully connected CRF [37, 102, 7, 64]. The general architecture of a FCN includes a sequence of convolution and downsampling layers (*encoder*), which extract multi–scale features, followed by a sequence of deconvolution layers (*decoder*), which produce a high–resolution segmentation (or “prediction”). Skip layers are often added, providing shortcut links from an encoder layer to its corresponding decoder layer. The role of skip layers is to provide well-localized information to a decoder layer, in addition to the semantic-rich but poorly resolved information from the prior decoder layer. In this way, skip layers enable good pixel-level localization while facilitating gradient flow during training. Similar architectures have been used in various applications such as text segmentation [63, 100, 64, 92], edge detection [89], face segmentation [61], and scene parsing [101]. Although these algorithms could be used for the foreground segmentation component of a content removal system, unavoidable inaccuracies are liable to dramatically decrease the quality of the recovered background region.

Image inpainting [4] has a long history. The goal of inpainting is to fill in a missing region with realistic image content. A variety of inpainting methods have been proposed, including those based on prior image statistics [69, 105], and those based on CNNs [67, 98]. More recently, outstanding results have been demonstrated with the use of GANs to inpaint even large missing areas [94, 91, 60, 96]. While appropriate for certain domains (e.g. face inpainting), methods in this category often suffer from serious limitations, including the requirement that the missing region have fixed size and shape.

All of the inpainting methods mentioned above assume that the corrupted region mask is known (typically as provided by the user). This limits their scope of application, as in most cases, this mask is unavailable. This shortcoming is addressed by blind inpainting algorithms [88, 43], which do not need access to the foreground mask. However, prior blind inpainting work has been demonstrated only for very simple cases, with constant-valued foreground occupying a small area of the image.

5.3 Proposed Algorithm

Our system for automatic semantic content removal comprises an encoder-decoder network with two decoder branches, tasked with predicting a segmentation mask (*dec-seg*) and a background image (*dec-fill*) in a single forward pass. Neglect nodes (an original feature of this architecture) link the two decoder branches and the encoder at various levels. The network is trained along with a discriminator network in an adversarial scheme, in order to foster realistic background image synthesis.

We assume in this work that the foreground region to be removed occupies a large portion of the image (or, equivalently, that the image is cropped such that the foreground region takes most of the cropped region). In practice, this can be obtained using a standard object detector. Note that high accuracy of the (rectangular) detector is not required. In our experiments, the margin between the contour of the foreground region and the edges of the image was let to vary between 0 (foreground touching the image edge) to half the size of the foreground mask.

5.3.1 Loss Function

Following the terminology of GANs, the output z_p of *dec-seg* and y_p of *dec-fill* for an input image x are taken to represent the output of a generator $G(x)$. The generator is trained with a dual task: ensuring that z_p and y_p are similar to the ground–truth (z_g and y_g), and deceiving the discriminator $D(x, y)$, which is concurrently trained to separate y_p from y_g given x. The cost function L_G for the generator combines the conditional GAN loss with a linear combination of the L_1 distances between prediction and ground–truth for segmentation and inpainted background:

$$L_G = \mathbb{E}[\log(1 - D(x, y_p))] + \lambda_f \mathbb{E}[||y_g - y_p||_1] + \lambda_s \mathbb{E}[||z_g - z_p||_1] \tag{5.1}$$

The discriminator D is trained to minimize the following discriminator loss L_D:

$$L_D = -(\mathbb{E}[\log D(x, y_g)] + \mathbb{E}[\log(1 - D(x, y_p))]) \tag{5.2}$$

5.3.2 Network Architecture

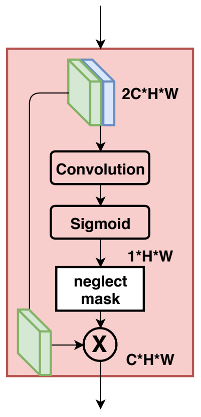
5.3.2.1 Generator

Segmentation and background infilling is generated in a single pass by an encoder–decoder network architecture, with multiple encoder layers generating multi-scale features at decreasing resolution, and two parallel decoder branches (*dec-seg* and *dec-fill*) producing high–resolution output starting from low–resolution features and higher–resolution data from skip layers. Each encoder stage consists of a convolution layer with kernel of size 4×4, stride 2, followed by instance normalization [79] and ReLU. Each stage of *dec-seg* contains a deconvolution (transpose convolution) layer (kernel of 4×4, stride 2), followed by instance normalization and ReLU. *dec-fill* replaces each deconvolution layers with a nearest–neighbor upsampling layer followed by a convolution layer (kernel sized 3×3, stride 1). This strategy, originally proposed by Odena *et al.* [58] to reduce checkerboard artifacts, was found to be very useful in our experiments (see Sec. 5.4.3). The total number of convolution kernels at the i-th encoder layer is $\min(2^{i-1} \times 64, 512)$. The number of deconvolution kernels at the i-th *dec-seg* layer or convolution kernels at the i-th *dec-fill* layer are the same as the number of kernels at the $(i-1)$-th encoder layer. The output of the first *dec-seg* layer and *dec-fill* layer has one channel (foreground segmentation) and three channels (recovered background image) respectively. All ReLU layers in the encoder are leaky, with slope of 0.2. In the *dec-seg* branch, standard skip layers are added. More precisely, following the layer indexing in Fig. 5.1, the input of the i-th layer of *dec-seg* is a concatenation of the output of the $(i + 1)$-th layer in the same branch and of the output of the i-th encoder layer (except for the 7-th decoder layer, which only receives input from the 7-th encoder layer.) As mentioned earlier, skip layers ensure good segmentation localization.

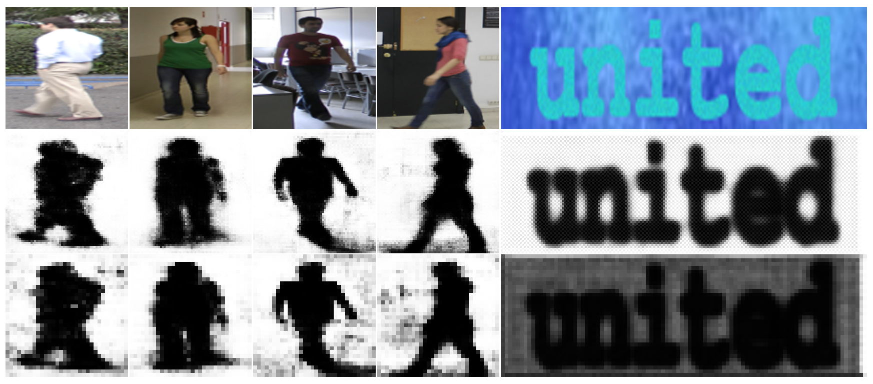
The layers of *dec-fill* also receive information from equi-scale encoder layers, but this information is modulated by *neglect masks* generated by neglect nodes. Specifically, the i-th neglect node receives in input data from the i-th encoder layer, concatenated with data from the $(i + 1)$-th *dec-seg* layer (note that this is the same as the input to the i-th *dec-seg* layer). A 1×1 convolution, followed by a sigmoid, produces a neglect mask (an image with values between 0 and 1). The neglect mask modulates (by pixel multiplication) the content of the i-th encoder layer, before it is concatenated with the output of the $(i + 1)$-th *dec-fill* layer and fed to the i-th *dec-fill* layer. The process is shown in Fig. 5.2 (a). In practice, neglect nodes provide *dec-fill* with information about which areas of the image should be erased and infilled, while preserving content elsewhere. Visual inspection of the neglect masks shows that they faithfully identify the portion of the image to be removed at various scales (see *e.g.*Fig. 5.2).

5.3.2.2 Discriminator

The input to the discriminator is the concatenation of the input image x and of the predicted background y_p or background ground–truth y_g. The structure of the



(a)



(b)

Figure 5.2: (a) Neglect node architecture. Green blocks: encoder output. Blue block: output of previous *dec-seg* layer. (b) Top row: input images. Second and third row: associated neglect masks generated by the neglect node in layer 1 and 2, respectively.

discriminator is the same as the first 5 encoder layers of the generator, but its output undergoes a 1×1 convolution layer followed by a sigmoid function. In the case of 128×128 input dimension, the output size is 4×4, with values between 0 and 1, representing the decoder's confidence that the corresponding region is realistic. The average of these 16 values forms the final discriminator output.

5.4 Experiments

5.4.1 Datasets

Training our model requires, for each image, three pieces of information: the original background image (for *dec-fill*); the foreground region (mask) to be removed (for *dec-seg*); and the composite image. Given that this type of rich information is not available in existing datasets, we built our own training and test sets. Specifically, we considered two different datasets for our experiments: one with synthetic text overimposed on regular images, and one with real images of pedestrians.

5.4.1.1 Text Dataset

Images in this dataset are generated by pasting text (generated synthetically) onto real background images. In this way, we have direct access to all necessary data (foreground, background, and composite image). Text images come from two resources: (1) the word synthesis engine of [64], which was used to generate 50K word images, along with the ground–truth associated segmentation masks; (2) the ICDAR 2013 dataset [1], which provides pixel-level text stroke labels, allowing us to extract 1850 real text regions. Random geometry transformations and color jittering was used to augment the real text, obtaining 50K more word images. Given a sample from the 100K word image pool, a similarly sized background image patch was cropped at random positions from images randomly picked from the MS COCO dataset[41]. More specifically, the background images for our training and validation set come from the training portion of the MS COCO dataset, while the background images for our test set come from the testing portion of the MS COCO dataset. This ensures that the training and testing sets do not share background images. In total, the training, validation and testing portions of our synthetic text dataset contain 100K, 15K and 15K images, respectively.

5.4.1.2 Pedestrian Dataset

This is built from the LASIESTA dataset[11], which contains several video sequences taken from a fixed camera with moving persons in the scene. LASIESTA provides ground–truth pedestrian segmentation for each frame in the videos. In this case, ground–truth background (which is occluded by a person at a given frame) can be found from neighboring frames, after the person has moved away. The foreground map is set to be equal to the segmentation mask provided with the dataset. We randomly selected 15 out of 17 video sequences for training, leaving the rest for testing. A sample of 1821 training images was augmented to 45K images via random cropping and color jittering. The test data set contains 198 images.

5.4.2 Implementation Details

Our system was implemented using Tensorflow and runs on a desktop (3.3Ghz 6-core CPU, 32G RAM, Titan XP GPU). The model was trained with input images resized to 384×128 (for the text dataset images) or 128×128 (for the pedestrian dataset images.) Adam solver was used during training, with learning rate set to 0.0001, momentum terms set to $\beta_1 = 0.5$ and $\beta_2 = 0.999$, and batch size equal to 8. We set $\lambda_f = \lambda_s = 100$ in the generator loss (5.1), and followed the standard GAN training strategy [22]. Training is alternated between discriminator (D) and generator (G). Note that the adversarial term for the cost L_G in (5.1) was changed to $(-\log(D(x, y_p)))$ (rather than $\log(1-D(x, y_p))$) for better numerical stability, as suggested by Goodfellow *et al.* [22]. When training D, the learning rate was divided by 2.

Method	Text			Pedestrian			Time (s)
	L_1	PSNR	SSIM	L_1	PSNR	SSIM	
Exemplar (z_p)	5.44%	16.563	0.553	4.11%	17.99	0.74	15.6
Exemplar (z_g)	5.46%	16.769	0.554	3.87%	18.36	0.77	
Contextual (z_p)	3.59%	19.788	0.752	3.41%	20.916	0.879	0.23
Contextual (z_g)	3.28%	20.170	0.779	2.64%	21.997	0.891	
EPLL (z_p)	2.00%	19.123	0.732	2.9%	16.143	0.780	53.5
EPLL (z_g)	1.87%	24.417	0.823	2.61%	16.852	0.800	
IRCNN (z_p)	2.62%	21.282	0.773	2.87%	19.117	0.870	6.67
IRCNN (z_g)	**1.79**%	**25.767**	0.835	**2.39**%	20.225	0.884	
Baseline	2.07%	24.188	0.811	3.06%	23.064	0.883	0.011
Ours (deconv)	1.91%	24.879	0.831	2.63%	23.612	0.904	0.018
Ours (nn+conv)	1.85%	25.300	**0.845**	2.55%	**23.877**	**0.918**	0.018

Table 5.1: Quantitative comparison of our method against other state-of-the-art image inpainting algorithms (Exemplar [10], Contextual [96], EPLL[105], and IRCNN[98]). Competing inpainting algorithms are fed with a segmentation mask, either predicted by our algorithm (z_p), or ground–truth (z_g). The difference between the original and reconstructed background image is measured using L_1 distance, PSNR (in dB, higher is better) and SSIM (higher is better)[85]. Time measurements refer to a 128×128 input image.

5.4.3 Ablation Study

5.4.3.1 Baseline

In order to validate the effectiveness of the two-branches decoder architecture and of the neglect layers, we compared our result against a simple baseline structure. This baseline structure is made by the encoder and the *dec-fill* decoding branch, without input from the neglect nodes, but with skip layers from the encoder. This is very very similar to the architecture proposed by Isola *et al.* [30]. Tab. 5.1 shows that our method consistently outperforms the baseline structure with both datasets and under all three evaluation metrics considered (L_1 residual, PSNR, SSIM [85]). This shows that explicit estimation of the segmentation mask, along with bypass input from the encoder modulated by the neglect mask, facilitates realistic background image synthesis. An example comparing the result of text removal and of inpainting using the full system and the baseline is shown in the first row of Fig. 5.3.

5.4.3.2 Deconvolution vs. upsampling + convolution

Deconvolution (or transpose convolution) is a standard approach for generating higher resolution images from coarse level features [65, 15, 30]. A problem with this technique is that it may produce visible checkerboard artifacts, which are due to "uneven overlapping" during the deconvolution process, especially when the kernel size is not divisible by the stride. Researchers [58] have found that by replacing deconvolution with nearest neighbor upsampling followed by convolution, these artifacts can be significantly reduced. In our experiments, we compared the results using these two techniques (see Fig. 5.3, second row). Specifically, upsampling + convolution was implemented using a kernel sized 3×3 with stride 1 (as described earlier in Sec. 5.3.2), while deconvolution was implemented by a kernel with size of 4×4 and stride of 2. Even though the deconvolution kernel side is divisible by the stride, checkerboard artifacts are still visible in most cases using deconvolution. These artifacts do not appear using upsampling + convolution, which also achieves better quantitative results as shown in Tab. 5.1.

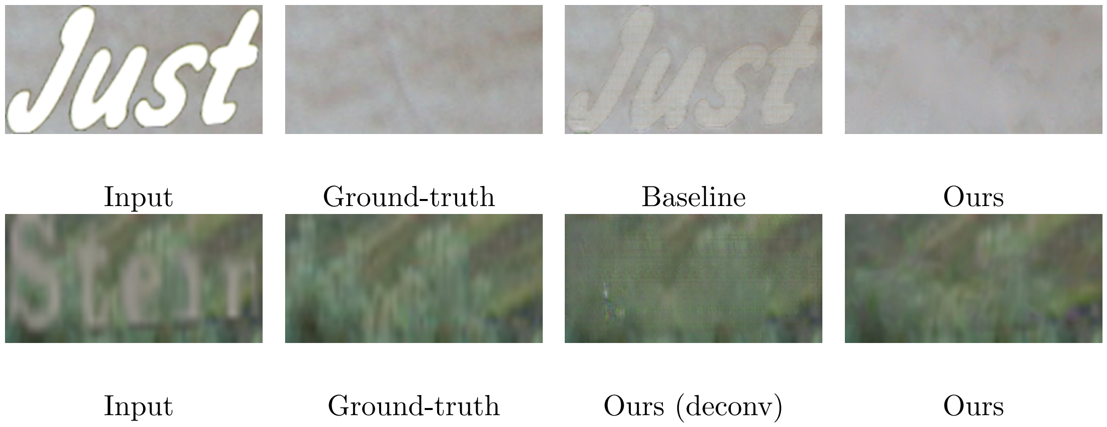


Figure 5.3: Experimental comparison between the baseline and our architecture (top), and between our architecture using deconvolution and using upsampling+convolution in *dec-fill* (bottom). See Sec. 5.4.3.

5.4.4 Comparative Results

Due to the lack of directly comparable methods for automatic content removal, we contrasted our technique with other state-of-the-art image inpainting algorithms, which were provided with a foreground mask. More specifically, we considered two setting for the foreground mask fed to these algorithms: (1) the segmentation mask obtained as a byproduct from our algorithm (z_p), and (2) the ground–truth mask (z_g). Note that the latter is a best-case scenario for the competing algorithms: our system never accessed this mask. In both cases, the masks were slightly dilated to ensure that the whole foreground region was covered.

Tab. 5.1 shows comparative results with two legacy (but still widely used) inpainting techniques (Exemplar [10] and EPLL[105]), as well as with two more recent CNN-based algorithms (IRCNN[98] and Contextual [96]). When fed with the z_p mask (setting (1)), all competing algorithms produced substantially inferior results with respect to ours under all metrics considered. Even when fed with the (unobservable) ground–truth mask z_g (setting (2)), these algorithms generally performed worse than our system (except for IRCNN, which gave better results than ours, under some of the metrics). We should stress that, unlike the competing techniques, our system *does not* receive an externally produced foreground map. Note also that our algorithm is faster (often by several orders of magnitude) than the competing techniques.

Fig. 5.4 shows comparative examples of results using our system, IRCNN, and Contextual (where the last two were fed with the ground–truth foreground mask, z_g). Note that, even when provided with the "ideal" mask, the visual quality of the results using these competing methods is generally inferior to that obtained with our content removal technique. The result of IRCNN, which is very similar to the result of EPLL, is clearly oversmoothed. This makes the object boundary visible due to the lack of high frequency details in the filled-in region. We also noted that this algorithms cannot cope well with large foreground masks, as can be seen in the last two columns of Fig. 5.4

(pedestrian dataset). Contextual [96] does a better job at recovering texture, thanks to its ability to explicitly utilize surrounding image features in its generative model. Yet, we found that our method is often better at completely removing foreground objects. Part of the foreground's boundary is still visible in Contextual's reconstructed background region. Furthermore, the quality Contextual's reconstruction drops significantly when the foreground region reaches the border of the image. This problem is not observed with our method.

Fig. 5.4 also reveals an interesting (and unexpected) feature of our system. As can be noted in the last two columns, the shadow cast by the person was removed along with the image of the person. Note that the system was *not* trained to detect shadows: the foreground mask only outlined the contour of the person. The most likely reason why the algorithm removed the shadow region is that the background images in the training set data (which, as mentioned in Sec. 5.4.1, were obtained from frames that did not contain the person) did not contain cast shadows of this type. The system thus decided to synthesize a shadowless image, doubling up as a shadow remover.

5.5 Conclusion

We have presented the first automatic content removal and impainting system that can work with widely different types and sizes of the foreground to be removed and infilled. Comparison with other state-of-the-art inpainting algorithms (which, unlike are system, need an externally provided foreground mask), along with the ablation study,

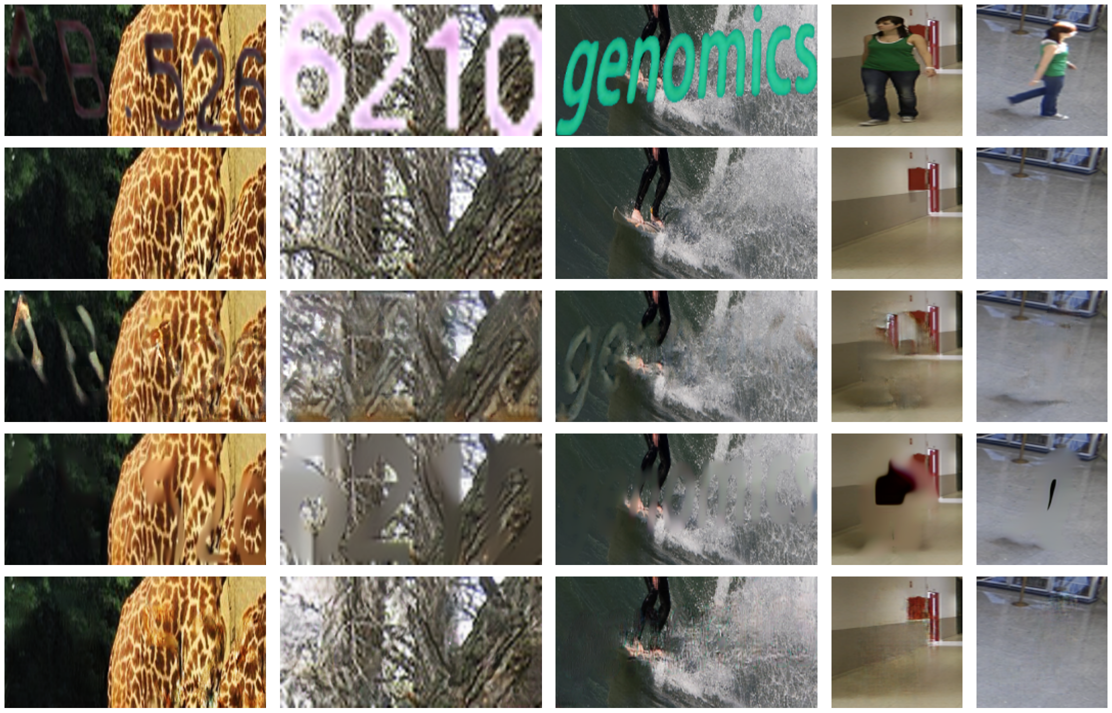


Figure 5.4: Sample inpainting results using Contextual [96] (third row), IRCNN [98] (fourth row), and our system (last row). Contextual and IRCNN were fed with the ground–truth segmentation mask, while our system automatically extracted and inpainted the foreground. Top row: input image. Second row: ground-truth background image.

show that our strategy of joint segmentation and inpainting provides superior results in most cases, at a lower computational cost. Future work will extend this technique to more complex scenarios such as wider ranges of foreground region sizes and transparent foreground.

Chapter 6

Conclusion

In this thesis, I introduce two text detection system, a novel text stroke segmentation algorithm with state-of-the-art performance, and a new encoder-decoder network architecture combined with an adversarial trained discriminator for automatic text removal.

Our first proposed text detection algorithm shares similar general structure (MSER computation, CNN classification, text line grouping) with other successful methods. Our contributions include a carefully designed strategy for thinning out MSERs that significantly reduces the computation cost with a minor drop in performance; a method for mining character-level training samples from word-level labeling; as well as the use of binarized patches for region classification. A more robust and efficient text spotting system is also presented, which is formed by two cascaded networks, TextSegNet and WordDetNet. The TextSegNet is trained to segment text blocks in a image, each text block may contain one or more words. The WordDetNet is trained to predict the coordinates of each word, given each text block as input. The first segmentation network determines regions of interest to be processed in detail by the second detection network.

To segment text stroke from its background, I present a FCN based algorithm. The output of FCN is further refined by a fully connected CRF using a novel kernel definition, which combines the traditional bilateral kernel with text-specific stroke width information. Our method outperforms state-of-the-art algorithms on standard benchmarks while being more computationally efficient.

Last but not least, I present an automatic text removal algorithm with promising results on challenging cases. The proposed algorithm is able to jointly predict the foreground text stroke segmentation and he background image. In order to achieve this, a new encoder-decoder architecture is proposed with two parallel and interconnected decoder branches, one focus on foreground segmentation and the other focus on background recovery. The neglect nodes, which is an original contribution of our work and the key to the good result, is used to determine which information should be used for background synthesis.

Bibliography

[1] Icdar focused scene text dataset. `http://rrc.cvc.uab.es/?ch=2`.

[2] Icdar incidental scene text dataset. `http://rrc.cvc.uab.es/?ch=4&com=introduction`.

[3] Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan B Goldman. Patchmatch: A randomized correspondence algorithm for structural image editing. *ACM Trans. Graph.*, 28(3):24–1, 2009.

[4] Marcelo Bertalmio, Guillermo Sapiro, Vincent Caselles, and Coloma Ballester. Image inpainting. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 417–424. ACM Press/Addison-Wesley Publishing Co., 2000.

[5] Michal Busta, Lukas Neumann, and Jiri Matas. Fastext: Efficient unconstrained scene text detector. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1206–1214, 2015.

[6] Huizhong Chen, Sam S Tsai, Georg Schroth, David M Chen, Radek Grzeszczuk,

and Bernd 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.

[7] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *arXiv preprint arXiv:1606.00915*, 2016.

[8] Xiangrong Chen and Alan 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.

[9] Adam Coates, Blake Carpenter, Carl Case, Sanjeev Satheesh, Bipin Suresh, Tao Wang, David J Wu, and Andrew 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.

[10] Antonio Criminisi, Patrick Pérez, and Kentaro Toyama. Region filling and object removal by exemplar-based image inpainting. *IEEE Transactions on image processing*, 13(9):1200–1212, 2004.

[11] Carlos Cuevas, Eva María Yáñez, and Narciso García. Labeled dataset for integral evaluation of moving object detection algorithms: Lasiesta. *Computer Vision and Image Understanding*, 152:103–117, 2016.

[12] M. H. Daraei, A. Vu, and R. Manduchi. Region segmentation using lidar and camera. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–6, 2017.

[13] M. Hossein Daraei, Ahn Vu, and Roberto Manduchi. Velocity and shape from tightly-coupled lidar and camera. In *2017 IEEE Intelligent Vehicles Symposium*, 06/2017 2017.

[14] Piotr Dollár, Ron Appel, Serge Belongie, and Pietro Perona. Fast feature pyramids for object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(8):1532–1545, 2014.

[15] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning. *arXiv preprint arXiv:1605.09782*, 2016.

[16] Boris Epshtein, Eyal Ofek, and Yonatan 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.

[17] Jacqueline Feild and Erik Learned-Miller. Scene text recognition with bilateral regression. *Department of Computer Science, University of Massachusetts Amherst, Tech. Rep. UM-CS-2012-021*, 2012.

[18] Victor Fragoso, Steffen Gauglitz, Shane Zamora, Jim Kleban, and Matthew Turk. Translatar: A mobile augmented reality translator. In *Applications of Computer Vision (WACV), 2011 IEEE Workshop on*, pages 497–502. IEEE, 2011.

[19] Andrea Frome, German Cheung, Ahmad Abdulkader, Marco Zennaro, Bo Wu, Alessandro Bissacco, Hartwig Adam, Hartmut Neven, and Luc Vincent. Large-scale privacy protection in google street view. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 2373–2380. IEEE, 2009.

[20] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[21] Rafael C Gonzalez and Juan A Herrera. Apparatus for reading a license plate, March 1989. US Patent 4,817,166.

[22] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[23] Ankush Gupta, Andrea Vedaldi, and Andrew Zisserman. Synthetic data for text localisation in natural images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2315–2324, 2016.

[24] Shehzad Muhammad Hanif and Lionel 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.

[25] Tong He, Weilin Huang, Yu Qiao, and Jian Yao. Text-attentional convolutional

neural network for scene text detection. *IEEE Transactions on Image Processing*, 25(6):2529–2541, 2016.

[26] Nicholas R Howe. A laplacian energy for document binarization. In *Document Analysis and Recognition (ICDAR), 2011 International Conference on*, pages 6–10. IEEE, 2011.

[27] Weilin Huang, Zhe Lin, Jianchao Yang, and Jue 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.

[28] Weilin Huang, Yu Qiao, and Xiaoou Tang. Robust scene text detection with convolution neural network induced mser trees. In *ECCV 2014*, pages 497–511. Springer, 2014.

[29] Alex D Hwang and Eli Peli. An augmented-reality edge enhancement application for google glass. *Optometry and vision science: official publication of the American Academy of Optometry*, 91(8):1021, 2014.

[30] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. *arXiv preprint*, 2017.

[31] Max Jaderberg, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Reading text in the wild with convolutional neural networks. *International Journal of Computer Vision*, pages 1–20, 2014.

[32] Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. Deep features for text spotting. In *ECCV 2014*, pages 512–528. Springer, 2014.

[33] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the ACM International Conference on Multimedia*, pages 675–678. ACM, 2014.

[34] Dimosthenis Karatzas, Lluis Gomez-Bigorda, Anguelos Nicolaou, Suman Ghosh, Andrew Bagdanov, Masakazu Iwamura, Jiri Matas, Lukas Neumann, Vijay Ramaseshan Chandrasekhar, Shijian Lu, et al. Icdar 2015 competition on robust reading. In *Document Analysis and Recognition (ICDAR), 2015 13th International Conference on*, pages 1156–1160. IEEE, 2015.

[35] Dimosthenis Karatzas, Faisal Shafait, Seiichi Uchida, Mikio Iwamura, Lluis Gomez i Bigorda, Sergi Robles Mestre, Jordi Mas, David Fernandez Mota, Jon Almazan Almazan, and Lluis-Pere de las Heras. Icdar 2013 robust reading competition. In *12th International Conference on Document Analysis and Recognition (ICDAR)*, pages 1484–1493. IEEE, 2013.

[36] Vladlen Koltun. Efficient inference in fully connected crfs with gaussian edge potentials. *Advances in Neural Information Processing Systems (NIPS)*, 2011.

[37] Philipp Krähenbühl and Vladlen Koltun. Efficient inference in fully connected

crfs with gaussian edge potentials. In *Advances in neural information processing systems*, pages 109–117, 2011.

[38] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[39] Deepak Kumar, MN Prasad, and AG Ramakrishnan. Benchmarking recognition results on camera captured word image data sets. In *Proceeding of the workshop on Document Analysis and Recognition*, pages 100–107. ACM, 2012.

[40] Minghui Liao, Baoguang Shi, Xiang Bai, Xinggang Wang, and Wenyu Liu. Textboxes: A fast text detector with a single deep neural network. *arXiv preprint arXiv:1611.06779*, 2016.

[41] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision*, pages 740–755. Springer, 2014.

[42] Xiaoqing Liu and Jagath Samarabandu. Multiscale edge-based text extraction from complex images. In *Multimedia and Expo, 2006 IEEE International Conference on*, pages 1721–1724. IEEE, 2006.

[43] Yang Liu, Jinshan Pan, and Zhixun Su. Deep blind image inpainting. *arXiv preprint arXiv:1712.09078*, 2017.

[44] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[45] Shijian Lu, Tao Chen, Shangxuan Tian, Joo-Hwee Lim, and Chew-Lim Tan. Scene text extraction based on edges and support vector regression. *International Journal on Document Analysis and Recognition (IJDAR)*, 18(2):125–135, 2015.

[46] Simon M Lucas, Alex Panaretos, Luis Sosa, Anthony Tang, Shirley Wong, and Robert Young. Icdar 2003 robust reading competitions. In *Document Analysis and Recognition, 2003. Proceedings. Seventh International Conference on*, pages 682–687. IEEE, 2003.

[47] Céline Mancas-Thillou and Bernard Gosselin. Color text extraction with selective metric-based clustering. *Computer Vision and Image Understanding*, 107(1):97–107, 2007.

[48] M Mann. Reading machine spells out loud. *Popular Science*, 154:125–7, 1949.

[49] Jiri Matas, Ondrej Chum, Martin Urban, and Tomás Pajdla. Robust wide-baseline stereo from maximally stable extremal regions. *Image and vision computing*, 22(10):761–767, 2004.

[50] Sergey Milyaev, Olga Barinova, Tatiana Novikova, Pushmeet Kohli, and Victor Lempitsky. Image binarization for end-to-end text understanding in natural im-

ages. In *Document Analysis and Recognition (ICDAR), 2013 12th International Conference on*, pages 128–132. IEEE, 2013.

[51] Anand Mishra, Karteek Alahari, and CV Jawahar. An mrf model for binarization of natural scene text. In *Document Analysis and Recognition (ICDAR), 2011 International Conference on*, pages 11–16. IEEE, 2011.

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

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

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

[55] Lukáš Neumann and Jiří Matas. Efficient scene text localization and recognition with local character refinement. *arXiv preprint arXiv:1504.03522*, 2015.

[56] Wayne Niblack. *An introduction to digital image processing*. Strandberg Publishing Company, 1985.

[57] David Nistér and Henrik Stewénius. Linear time maximally stable extremal regions. In *ECCV 2008*, pages 183–196. Springer, 2008.

[58] Augustus Odena, Vincent Dumoulin, and Chris Olah. Deconvolution and checkerboard artifacts. *Distill*, 1(10):e3, 2016.

[59] Nobuyuki Otsu. A threshold selection method from gray-level histograms. *IEEE transactions on systems, man, and cybernetics*, 9(1):62–66, 1979.

[60] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2536–2544, 2016.

[61] Siyang Qin, Seongdo Kim, and Roberto Manduchi. Automatic skin and hair masking using fully convolutional networks. In *Proceedings of the IEEE International Conference on Multimedia and Expo*, 2017.

[62] Siyang Qin and Roberto Manduchi. A fast and robust text spotter. In *Proceedings of the IEEE Winter Conference on Applications of Computer Vision*, 2016.

[63] Siyang Qin and Roberto Manduchi. Cascaded segmentation-detection networks for word-level text spotting. *arXiv preprint arXiv:1704.00834*, 2017.

[64] Siyang Qin, Peng Ren, and Roberto Kim, Seongdo Manduchi. Robust and accurate text stroke segmentation. In *Proceedings of the IEEE Winter Conference on Applications of Computer Vision*, 2018.

[65] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation

learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[66] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 779–788, 2016.

[67] Jimmy SJ Ren, Li Xu, Qiong Yan, and Wenxiu Sun. Shepard convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 901–909, 2015.

[68] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[69] Stefan Roth and Michael J Black. Fields of experts: A framework for learning image priors. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 860–867. IEEE, 2005.

[70] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. Grabcut: Interactive foreground extraction using iterated graph cuts. In *ACM transactions on graphics (TOG)*, volume 23, pages 309–314. ACM, 2004.

[71] Jaakko Sauvola and Matti Pietikäinen. Adaptive document image binarization. *Pattern recognition*, 33(2):225–236, 2000.

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

[73] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[74] Bolan Su, Shijian Lu, and Chew Lim Tan. Binarization of historical document images using the local maximum and minimum. In *Proceedings of the 9th IAPR International Workshop on Document Analysis Systems*, pages 159–166. ACM, 2010.

[75] Shangxuan Tian, Shijian Lu, Bolan Su, and Chew Lim Tan. Scene text segmentation with multi-level maximally stable extremal regions. In *Pattern Recognition (ICPR), 2014 22nd International Conference on*, pages 2703–2708. IEEE, 2014.

[76] Shangxuan Tian, Shijian Lu, Bolan Su, and Chew Lim Tan. Robust text segmentation using graph cut. In *Document Analysis and Recognition (ICDAR), 2015 13th International Conference on*, pages 331–335. IEEE, 2015.

[77] Shangxuan Tian, Yifeng Pan, Chang Huang, Shijian Lu, Kai Yu, and Chew Lim Tan. Text flow: A unified text detection system in natural scene images. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4651–4659, 2015.

[78] Zhi Tian, Weilin Huang, Tong He, Pan He, and Yu Qiao. Detecting text in natural image with connectionist text proposal network. In *European Conference on Computer Vision*, pages 56–72. Springer, 2016.

[79] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint*, 2016.

[80] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 6000–6010, 2017.

[81] Andreas Veit, Tomas Matera, Lukas Neumann, Jiri Matas, and Serge Belongie. Coco-text: Dataset and benchmark for text detection and recognition in natural images. *arXiv preprint arXiv:1601.07140*, 2016.

[82] Kai Wang, Boris Babenko, and Serge Belongie. End-to-end scene text recognition. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1457–1464. IEEE, 2011.

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

[84] Xiufei Wang, Lei Huang, and Changping Liu. A novel method for embedded text segmentation based on stroke and color. In *Document Analysis and Recognition (ICDAR), 2011 International Conference on*, pages 151–155. IEEE, 2011.

[85] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[86] 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.

[87] Christian Wolf and Jean-Michel Jolion. Object count/area graphs for the evaluation of object detection and segmentation algorithms. *International Journal of Document Analysis and Recognition (IJDAR)*, 8(4):280–296, 2006.

[88] Junyuan Xie, Linli Xu, and Enhong Chen. Image denoising and inpainting with deep neural networks. In *Advances in neural information processing systems*, pages 341–349, 2012.

[89] Saining Xie and Zhuowen Tu. Holistically-nested edge detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1395–1403, 2015.

[90] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*, pages 2048–2057, 2015.

[91] Chao Yang, Xin Lu, Zhe Lin, Eli Shechtman, Oliver Wang, and Hao Li. High-resolution image inpainting using multi-scale neural patch synthesis. In *The IEEE*

Conference on Computer Vision and Pattern Recognition (CVPR), volume 1, page 3, 2017.

[92] Cong Yao, Xiang Bai, Nong Sang, Xinyu Zhou, Shuchang Zhou, and Zhimin Cao. Scene text detection via holistic, multi-channel prediction. *arXiv preprint arXiv:1606.09002*, 2016.

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

[94] Raymond A Yeh, Chen Chen, Teck Yian Lim, Alexander G Schwing, Mark Hasegawa-Johnson, and Minh N Do. Semantic image inpainting with deep generative models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5485–5493, 2017.

[95] Xu-Cheng Yin, Xuwang Yin, Kaizhu Huang, and Hong-Wei Hao. Robust text detection in natural scene images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(5):970–983, 2014.

[96] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S Huang. Generative image inpainting with contextual attention. *arXiv preprint arXiv:1801.07892*, 2018.

[97] Alessandro Zamberletti, Lucia Noce, and Ignazio Gallo. Text localization based

on fast feature pyramids and multi-resolution maximally stable extremal regions. In *ACCV 2014 Workshops*, pages 91–105. Springer, 2014.

[98] Kai Zhang, Wangmeng Zuo, Shuhang Gu, and Lei Zhang. Learning deep cnn denoiser prior for image restoration. *arXiv preprint*, 2017.

[99] Zheng Zhang, Wei Shen, Cong Yao, and Xiang 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.

[100] Zheng Zhang, Chengquan Zhang, Wei Shen, Cong Yao, Wenyu Liu, and Xiang Bai. Multi-oriented text detection with fully convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4159–4167, 2016.

[101] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 2881–2890, 2017.

[102] Shuai Zheng, Sadeep Jayasumana, Bernardino Romera-Paredes, Vibhav Vineet, Zhizhong Su, Dalong Du, Chang Huang, and Philip HS Torr. Conditional random fields as recurrent neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1529–1537, 2015.

[103] Yahan Zhou, Jacqueline Feild, Erik Learned-Miller, and Rui Wang. Scene text

segmentation via inverse rendering. In *Document Analysis and Recognition (ICDAR), 2013 12th International Conference on*, pages 457–461. IEEE, 2013.

[104] Siyu Zhu and Richard Zanibbi. A text detection system for natural scenes with convolutional feature learning and cascaded classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 625–632, 2016.

[105] Daniel Zoran and Yair Weiss. From learning models of natural image patches to whole image restoration. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 479–486. IEEE, 2011.