Synthetic Data for Text Localisation in Natural Images

Ankush Gupta   Andrea Vedaldi   Andrew Zisserman
Dept. of Engineering Science, University of Oxford
{ankush,vedaldi,az}@robots.ox.ac.uk

Abstract

In this paper we introduce a new method for text detection in natural images. The method comprises two contributions: First, a fast and scalable engine to generate synthetic images of text in clutter. This engine overlays synthetic text to existing background images in a natural way, accounting for the local 3D scene geometry. Second, we use the synthetic images to train a Fully-Convolutional Regression Network (FCRN) which efficiently performs text detection and bounding-box regression at all locations and multiple scales in an image. We discuss the relation of FCRN to the recently-introduced YOLO detector, as well as other end-to-end object detection systems based on deep learning. The resulting detection network significantly out performs current methods for text detection in natural images, achieving an F-measure of 84.2% on the standard ICDAR 2013 benchmark. Furthermore, it can process 15 images per second on a GPU.

1. Introduction

Text spotting, namely the ability to read text in natural scenes, is a highly-desirable feature in anthropocentric applications of computer vision. State-of-the-art systems such as [20] achieved their high text spotting performance by combining two simple but powerful insights. The first is that complex recognition pipelines that recognise text by explicitly combining recognition and detection of *individual* characters can be replaced by very powerful classifiers that directly map an image patch to words [13, 20]. The second is that these powerful classifiers can be learned by generating the required training data synthetically [19, 44].

While [20] successfully addressed the problem of recognising text *given an image patch containing a word*, the process of obtaining these patches remains suboptimal. The pipeline combines general purpose features such as HoG [6], EdgeBoxes [48] and Aggregate Channel Features [7] and brings in text specific (CNN) features only in the later stages, where patches are finally recognised as specific words. This state of affair is highly undesirable for two
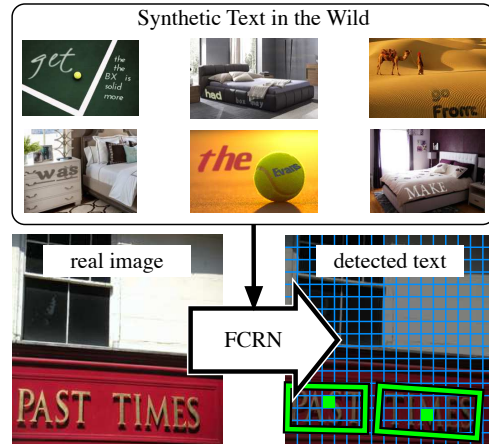

Figure 1. We propose a Fully-Convolutional Regression Network (FCRN) for high-performance text recognition in natural scenes (bottom) which detects text up to $45\times$ faster than the current state-of-the-art text detectors and with better accuracy. FCRN is trained without any manual annotation using a new dataset of synthetic text in the wild. The latter is obtained by automatically adding text to natural scenes in a manner compatible with the scene geometry (top).

reasons. First, the performance of the detection pipeline becomes the new bottleneck of text spotting: in [20] recognition accuracy for correctly cropped words is 98% whereas the end-to-end text spotting F-score is only 69% mainly due to incorrect and missed word region proposals. Second, the pipeline is slow and inelegant.

In this paper we propose improvements similar to [20] to the complementary problem of *text detection*. We make two key contributions. First, we propose a new method for generating synthetic images of text that *naturally blends text in existing natural scenes*, using off-the-shelf deep learning and segmentation techniques to align text to the geometry of a background image and respect scene boundaries. We use this method to automatically generate a new **synthetic dataset of text in cluttered conditions** (figure 1 (top) and section 2). This dataset, called *SynthText in the Wild* (figure 2), is suitable for training high-performance scene text detectors. The key difference with existing synthetic text datasets such as the one of [20] is that these only contains
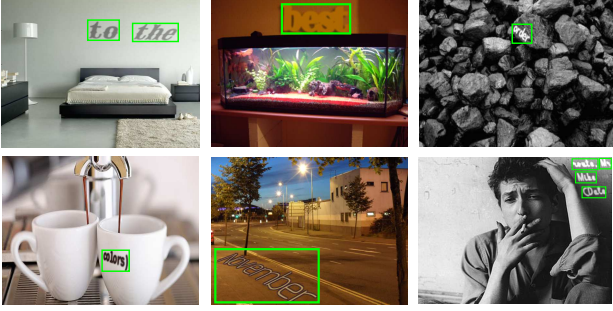
Figure 2. Sample images from our synthetically generated scene-text dataset. Ground-truth word-level axis-aligned bounding boxes are shown.

Dataset	# Images		# Words	
	Train	Test	Train	Test
ICDAR {11,13,15}	229	255	849	1095
SVT	100	249	257	647

Table 1. Size of publicly available text localisation datasets — ICDAR [23, 24, 39], the Street View Text (SVT) dataset [43]. Word numbers for the entry "ICDAR{11,13,15}" are from the IC-DAR15 Robust Reading Competition's Focused Scene Text Localisation dataset.

word-level image regions and are unsuitable for training detectors.

The second contribution is a **text detection deep architecture** which is both accurate and efficient (figure 1 (bottom) and section 3). We call this a *fully-convolutional regression network*. Similar to models such as the Fully-Convolutional Networks (FCN) for image segmentation, it performs prediction densely, at every image location. However, differently from FCN, the prediction is not just a class label (text/not text), but the parameters of a bounding box enclosing the word centred at that location. The latter idea is borrowed from the You Look Only Once (YOLO) technique of Redmon *et al*. [36], but with convolutional regressors with a significant boost to performance.

The new data and detector achieve state-of-the-art text detection performance on standard benchmark datasets (section 4) while being an order of magnitude faster than traditional text detectors at test time (up to 15 images per second on a GPU). We also demonstrate the importance of verisimilitude in the dataset by showing that if the detector is trained on images with words inserted synthetically that do *not* take account of the scene layout, then the detection performance is substantially inferior. Finally, due to the more accurate detection step, end-to-end word recognition is also improved once the new detector is swapped in for existing ones in state-of-the-art pipelines. Our findings are summarised in section 5.

1.1. Related Work

Object Detection with CNNs. Our text detection network draws primarily on Long *et al*.'s Fully-Convolutional network [31] and Redmon *et al*.'s YOLO image-grid based bounding-box regression network [36]. YOLO is part of a broad line of work on using CNN features for object category detection dating back to Girshick *et al*.'s Region-CNN (R-CNN) framework [12] combination of region proposals and CNN features. The R-CNN framework has three broad stages — (1) generating object proposals, (2) extracting CNN feature maps for each proposal, and (3) filtering the proposals through class specific SVMs. Jaderberg *et al*.'s text spotting method also uses a similar pipeline for detection [20]. Extracting feature maps for each region *independently* was identified as the bottleneck by Girshick *et al*. in Fast R-CNN [11]. They obtain $100\times$ speed-up over R-CNN by computing the CNN features once and pooling them locally for each proposal; they also streamline the last two stages of R-CNN into a single multi-task learning problem. This work exposed the region-proposal stage as the new bottleneck. Lenc *et al*. [29] drop the region proposal stage altogether and use a constant set of regions learnt through K-means clustering on the PASCAL VOC data. Ren *et al*. [37] also start from a fixed set of proposal, but refined them prior to detection by using a Region Proposal Network which shares weights with the later detection network and streamlines the multi-stage R-CNN framework.

Synthetic Data. Synthetic datasets provide detailed ground-truth annotations, and are cheap and scalable alternatives to annotating images manually. They have been widely used to learn large CNN models — Wang *et al*. [44] and Jaderberg *et al*. [19] use synthetic text images to train word-image recognition networks; Dosovitskiy *et al*. [9] use floating chair renderings to train dense optical flow regression networks. Detailed synthetic data has also been used to learn generative models — Dosovitskiy *et al*. [8] train inverted CNN models to render images of chairs, while Yildirim *et al*. [46] use deep CNN features trained on synthetic face renderings to regress pose parameters from face images.

Augmenting Single Images. There is a large body of work on inserting objects photo-realistically, and inferring 3D structure from single images — Karsch *et al*. [25] develop an impressive semi-automatic method to render objects with correct lighting and perspective; they infer the actual size of objects based on the technique of Criminisi *et al*. [5]. Hoiem *et al*. [15] categorise image regions into ground-plane, vertical plane or sky from a single image and use it to generate "pop-ups" by decomposing the image into planes [14]. Similarly, we too decompose a single image into local planar regions, but use instead the dense depth prediction of Liu *et al*. [30].

RGB   Depth   gPb-UCM Segmentation   Text Regions
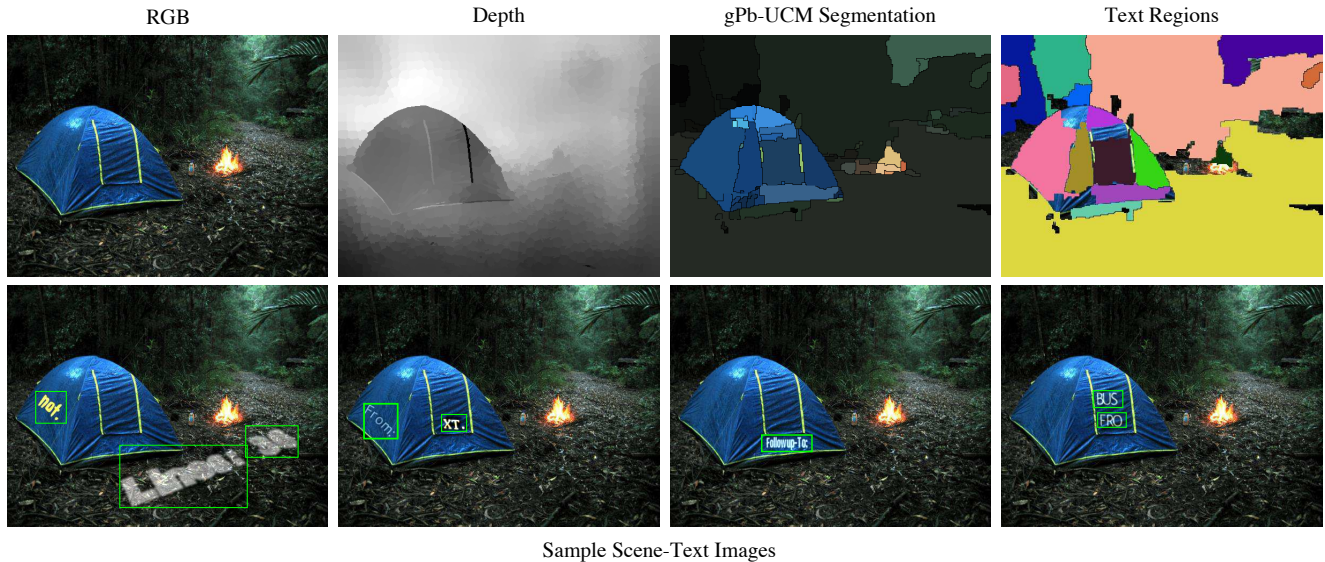
Sample Scene-Text Images

Figure 3. (Top, left to right): (1) RGB input image with no text instance. (2) Predicted dense depth map (darker regions are closer). (3) Colour and texture gPb-UCM segments. (4) Filtered regions: regions suitable for text are coloured randomly; those unsuitable retain their original image pixels. (Bottom): Four synthetic scene-text images with axis-aligned bounding-box annotations at the word level.

2. Synthetic Text in the Wild

Supervised training of large models such as deep CNNs, which contain millions of parameters, requires a very significant amount of labelled training data [26], which is expensive to obtain manually. Furthermore, as summarised in Table 1, publicly available text spotting or detection datasets are quite small. Such datasets are not only insufficient to train large CNN models, but also inadequate to represent the space of possible text variations in natural scenes — fonts, colours, sizes, positions. Hence, in this section we develop a synthetic text-scene image generation engine for building a large annotated dataset for text localisation.

Our synthetic engine (1) produces **realistic** scene-text images so that the trained models can generalise to real (non-synthetic) images, (2) is fully **automated** and, is (3) **fast**, which enables the generation of large quantities of data without supervision. The text generation pipeline can be summarised as follows (see also Figure 3). After acquiring suitable text and image samples (section 2.1), the image is segmented into contiguous regions based on local colour and texture cues [2], and a dense pixel-wise depth map is obtained using the CNN of [30] (section 2.2). Then, for each contiguous region a local surface normal is estimated. Next, a colour for text and, optionally, for its outline is chosen based on the region's colour (section 2.3). Finally, a text sample is rendered using a randomly selected font and transformed according to the local surface orientation; the text is blended into the scene using Poisson image editing [35]. Our engine takes about half a second to generate a new scene-text image.

This method is used to generate 800,000 scene-text im-

ages, each with multiple instances of words rendered in different styles as seen in Figure 2. The dataset is available at: http://www.robots.ox.ac.uk/~vgg/data/scenetext

2.1. Text and Image Sources

The synthetic text generation process starts by sampling some text and a background image. The text is extracted from the Newsgroup20 dataset [27] in three ways — words, lines (up to 3 lines) and paragraphs (up to 7 lines). Words are defined as tokens separated by whitespace characters, lines are delimited by the newline character. This is a rich dataset, with a natural distribution of English text interspersed with symbols, punctuation marks, nouns and numbers.

To favour variety, 8,000 background images are extracted from Google Image Search through queries related to different objects/scenes and indoor/outdoor and natural/artificial locales. To guarantee that all text occurrences are fully annotated, these images *must not contain text of their own* (a limitation of the Street View Text [43] is that annotations are not exhaustive). Hence, keywords which would recall a large amount of text in the images (*e.g.* "street-sign", "menu" etc.) are avoided; images containing text are discarded through manual inspection.

2.2. Segmentation and Geometry Estimation

In real images, text tends to be contained in well defined regions (e.g. a sign). We approximate this constraint by requiring text to be contained in regions characterised by a uniform colour and texture. This also prevents text from crossing strong image discontinuities, which is unlikely to
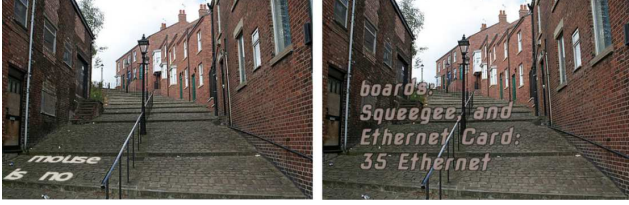
Figure 4. Local colour/texture sensitive placement. (Left) Example image from the Synthetic text dataset. Notice that the text is restricted within the boundaries of the step in the street. (Right) For comparison, the placement of text in this image does not respect the local region cues.

occur in practice. Regions are obtained by thresholding the gPb-UCM contour hierarchies [2] at 0.11 using the efficient graph-cut implementation of [3]. Figure 4 shows an example of text respecting local region cues.

In natural images, text tends to be painted on top of surfaces (e.g. a sign or a cup). In order to approximate a similar effect in our synthetic data, the text is perspectively transformed according to local surface normals. The normals are estimated automatically by first predicting a dense depth map using the CNN of [30] for the regions segmented above, and then fitting a planar facet to it using RANSAC [10].

Text is aligned to the estimated region orientations as follows: first, the image region contour is warped to a frontal-parallel view using the estimated plane normal; then, a rectangle is fitted to the fronto-parallel region; finally, the text is aligned to the larger side ("width") of this rectangle. When placing multiple instances of text in the same region, text masks are checked for collision against each other to avoid placing them on top of each other. Not all segmentation regions are suitable for text placement — regions should not be too small, have an extreme aspect ratio, or have surface normal orthogonal to the viewing direction; all such regions are filtered in this stage. Further, regions with too much texture are also filtered, where the degree of texture is measured by the strength of third derivatives in the RGB image.

Discussion. An alternative to using a CNN to estimate depth, which is an error prone process, is to use a dataset of RGBD images. We prefer to estimate an imperfect depth map instead because: (1) it allows essentially any scene type background image to be used, instead of only the ones for which RGBD data are available, and (2) because publicly available RGBD datasets such as NYUDv2 [40], B3DO [22], Sintel [4], and Make3D [38] have several limitations in our context: small size (1,500 images in NYUDv21, 400 frames in Make3D, and a small number of videos in B3DO and Sintel), low-resolution and motion blur, restriction to indoor images (in NYUDv2 and B3DO), and limited variability in the images for video-based datasets (B3DO and Sintel).

2.3. Text Rendering and Image Composition

Once the location and orientation of text has been decided, text is assigned a colour. The colour palette for text is learned from cropped word images in the IIIT5K word dataset [32]. Pixels in each cropped word images are partitioned into two sets using K-means, resulting in a colour pair, with one colour approximating the foreground (text) colour and the other the background. When rendering new text, the colour pair whose background colour matches the target image region the best (using L2-norm in the Lab colour space) is selected, and the corresponding foreground colour is used to render the text.

About 20% of the text instances are randomly chosen to have a border. The border colour is chosen to be either the same as foreground colour with its value channel increased or decreased, or is chosen to be the mean of the foreground and background colours.

To maintain the illumination gradient in the synthetic text image, we blend the text on to the base image using Poisson image editing [35], with the guidance field defined as in their equation (12). We solve this efficiently using the implementation provided by Raskar[1]

3. A Fast Text Detection Network

In this section we introduce our CNN architecture for text detection in natural scenes. While existing text detection pipelines combine several ad-hoc steps and are slow, we propose a detector which is highly accurate, fast, and trainable end-to-end.

Let $\mathbf{x}$ denote an image. The most common approach for CNN-based detection is to propose a number of image regions R that may contain the target object (text in our case), crop the image, and use a CNN $c = \phi(\mathrm{crop}_R(\mathbf{x})) \in \{0, 1\}$ to score them as correct or not. This approach, which has been popularised by R-CNN [12], works well but is slow as it entails evaluating the CNN thousands of times per image.

An alternative and much faster strategy for object detection is to construct a fixed field of predictors $(c, \mathbf{p}) = \phi_{uv}(\mathbf{x})$, each of which specialises in predicting the presence $c \in \mathbb{R}$ and pose $\mathbf{p} = (x - u, y - v, w, h)$ of an object around a specific image location (u, v). Here the pose parameters (x, y) and (w, h) denote respectively the location and size of a bounding box tightly enclosing the object. Each predictor ϕ_{uv} is tasked with predicting objects which occurs in some ball $(x, y) \in B_\rho(u, v)$ of the predictor location.

While this construction may sound abstract, it is actually a common one, implemented for example by Implicit Shape Models (ISM) [28] and Hough voting [16]. There a predictor ϕ_{uv} looks at a local image patch, centred at (u, v), and

---

[1]Fast Poisson image editing code available at: http://web.media.mit.edu/~raskar/photo/code.pdf based on Discrete Sine Transform.

tries to predict whether there is an object around (u, v), and where the object is located relative to it.

In this paper we propose an extreme variant of Hough voting, inspired by Fully-Convolutional Network (FCN) of Long *et al.* [31] and the You Look Only Once (YOLO) technique of Redmon *et al.* [36]. In ISM and Hough voting, individual predictions are aggregated across the image, in a voting scheme. YOLO is similar, but avoids voting and uses individual predictions directly; since this idea can accelerate detection, we adopt it here.

The other key conceptual difference between YOLO and Hough voting is that in Hough voting predictors $\phi_{uv}(\mathbf{x})$ are local and translation invariant, whereas in YOLO they are not: First, in YOLO each predictor is allowed to pool evidence from the whole image, not just an image patch centred at (u, v). Second, in YOLO predictors at different locations $(u, v) \neq (u', v')$ are different functions $\phi_{uv} \neq \phi_{u'v'}$ learned independently.

While YOLO's approach allows the method to pick up contextual information useful in detection of PASCAL or ImageNet objects, we found this unsuitable for smaller and more variable text occurrences. Instead, we propose here a method which is in between YOLO and Hough voting. As in YOLO, each detector $\phi_{uv}(\mathbf{x})$ still predicts directly object occurrences, without undergoing an expensive voting accumulation process; however, as in Hough voting, detectors $\phi_{uv}(\mathbf{x})$ are local and translation invariant, sharing parameters. We implement this field of translation-invariant and local predictors as the output of the last layer of a deep CNN, obtaining a *fully-convolutional regression network* (FCRN).

3.1. Architecture

This section describes the structure of the FCRN. First, we describe the first several layers of the architecture, which compute text-specific image features. Then, we describe the dense regression network built on top of these features and finally its application at multiple scales.

Single-scale features. Our architecture is inspired by VGG-16 [41], using several layers of small dense filters; however, we found that a much smaller model works just as well and more efficiently for text. The architecture comprises nine convolutional layers, each followed by the Rectified Linear Unit non-linearity, and, occasionally, by a max-pooling layer. All linear filters have a stride of 1 sample, and preserve the resolution of feature maps through zero padding. Max-pooling is performed over 2×2 windows with a stride of 2 samples, therefore halving the feature maps resolution.[2]

Class and bounding box prediction. The single-scale features terminate with a dense feature field. Given that there

are four downsampling max-pooling layers, the stride of these features is $\Delta = 16$ pixels, each containing 512 feature channels $\phi_{uv}^f(\mathbf{x})$ (we express uv in pixels for convenience).

Given the features $\phi_{uv}^f(\mathbf{x})$, we can now discuss the construction of the dense text predictors $\phi_{uv}(\mathbf{x}) = \phi_{uv}^r \circ \phi^f(\mathbf{x})$. These predictors are implemented as a further seven 5×5 linear filters (C-7-5×5) ϕ_{uv}^r, each regressing one of seven numbers: the object presence confidence c, and up to six object pose parameters $\mathbf{p} = (x - u, y - v, w, h, \cos\theta, \sin\theta)$ where x, y, w, h have been discussed before and θ is the bounding box rotation.

Hence, for an input image of size $H \times W$, we obtain a grid of $\frac{H}{\Delta} \times \frac{W}{\Delta}$ predictions, one each for an image cell of size $\Delta \times \Delta$ pixels. Each predictor is responsible for detecting a word if the word centre falls within the corresponding cell.[3] YOLO is similar but operates at about half this resolution; a denser predictor sampling is important to reduce collisions (multiple words falling in the same cell) and therefore to increase recall (since at most one word can be detected per cell). In practice, for a 224×224 image, we obtain 14×14 cells/predictors

Multi-scale detection. Limited receptive field of our convolutional filters prohibits detection of large text instances. Hence, we get the detections at multiple down-scaled versions of the input image and merge them through non-maximal suppression. In more detail, the input image is scaled down by factors $\{1, 1/2, 1/4, 1/8\}$ (scaling up is an overkill as the baseline features are already computed very densely). Then, the resulting detections are combined by suppressing those with a lower score than the score of an overlapping detection.

Training loss. We use a squared loss term for each of the $\frac{H}{\Delta} \times \frac{W}{\Delta} \times 7$ outputs of the CNN as in YOLO [36]. If a cell does not contain a ground-truth word, the loss ignores all parameters but c (text/no-text).

Comparison with YOLO. Our fully-convolutional regression network (FCRN) has $30\times$ less parameters than the YOLO network (which has $\sim 90\%$ of the parameters in the last two fully-connected layers). Due to its global nature, standard YOLO must be retrained for each image size, including multiple scales, further increasing the model size (while our model requires 44MB, YOLO would require 2GB). This makes YOLO not only harder to train, but also less efficient ($2\times$ slower that FCRN).

4. Evaluation

First, in section 4.1 we describe the text datasets on which we evaluate our model. Next, we evaluate our model on the text localisation task in section 4.2. In section 4.3, to investigate which components of the synthetic data generation pipeline are important, we perform detailed ablation

---

[2]The sequence of layers is as follows: 64 5×5 convolutional filters + ReLU (CR-64-5×5), max pooling (MP), CR-128-5×5, MP, CR128-3×3, CR-128-3×3-conv, MP, CR-256-3×3, CR-256-3×3, MP, CR-512-3×3, CR-512-3×3, CR-512-5×5.

[3]For regression, it was found beneficial to normalise the pose parameters as follows: $\bar{\mathbf{p}} = ((x - u)/\Delta, (y - v)/\Delta, w/W, h/H, \cos\theta, \sin\theta)$.

	PASCAL Eval												DetEval								
	IC11				IC13				SVT				IC11			IC13			SVT		
	F	P	R	R_M	F	P	R	R_M	F	P	R	R_M	F	P	R	F	P	R	F	P	R
Huang [17]	-	-	-	-	-	-	-	-	-	-	-	-	78	88	71	-	-	-	-	-	-
Jaderberg [20]	77.2	87.5	69.2	70.6	76.2	86.7	68.0	69.3	53.6	62.8	46.8	55.4	76.8	88.2	68.0	76.8	88.5	67.8	24.7	27.7	22.3
Jaderberg (trained on SynthText)	77.3	89.2	68.4	72.3	76.7	88.9	67.5	71.4	53.6	58.9	49.1	56.1	75.5	87.5	66.4	75.5	87.9	66.3	24.7	27.8	22.3
Neumann [33]	-	-	-	-	-	-	-	-	-	-	-	-	68.7	73.1	64.7	-	-	-	-	-	-
Neumann [34]	-	-	-	-	-	-	-	-	-	-	-	-	72.3	79.3	66.4	-	-	-	-	-	-
Zhang [47]	-	-	-	-	-	-	-	-	-	-	-	-	80	84	**76**	80	88	74	-	-	-
FCRN single-scale	60.6	78.8	49.2	49.2	61.0	77.7	48.9	48.9	45.6	50.9	41.2	41.2	64.5	81.9	53.2	64.3	81.3	53.1	31.4	**34.5**	28.9
FCRN multi-scale	70.0	78.4	63.2	64.6	69.5	78.1	62.6	67.0	46.2	47.0	45.4	53.0	73.0	77.9	68.9	73.4	80.3	67.7	**34.5**	29.9	**40.7**
FCRN + multi-filt	78.7	**95.3**	67.0	67.5	78.0	**94.8**	66.3	66.7	56.3	61.5	51.9	54.1	78.0	**94.5**	66.4	78.0	**94.8**	66.3	25.5	26.8	24.3
FCRNall + multi-filt	**84.7**	94.3	**76.9**	**79.6**	**84.2**	93.8	**76.4**	**79.6**	**62.4**	**65.1**	**59.9**	**75.0**	**82.3**	91.5	74.8	**83.0**	92.0	**75.5**	26.7	26.2	27.4

Table 2. Comparison with previous methods on text localisation. Precision (P) and Recall (R) at maximum F-measure (F) and the maximum recall (R_M) are reported.

experiments. In section 4.4, we use the results from our localisation model for end-to-end text spotting. We show substantial improvements over the state-of-the-art in both text localisation and end-to-end text spotting. Finally, in section 4.5 we discuss the speed-up gained by using our models for text localisation.

4.1. Datasets

We evaluate our text detection networks on standard benchmarks: *ICDAR* 2011, 2013 datasets [24, 39] and the Street View Text dataset [43]. These datasets are reviewed next and their statistics are given in Table 1.

SynthText in the Wild. This is a dataset of 800,000 training images generated using our synthetic engine from section 2. Each image has about ten word instances annotated with character and word-level bounding-boxes.

ICDAR Datasets. The *ICDAR* datasets (IC011, IC013) are obtained from the Robust Reading Challenges held in 2011 and 2013 respectively. They contain real world images of text on sign boards, books, posters and other objects with world-level axis-aligned bounding box annotations. The datasets largely contain the same images, but shuffle the test and training splits. We do not evaluate on the more recent *ICDAR* 2015 dataset as it is almost identical to the 2013 dataset.

Street View Text. This dataset, abbreviated *SVT*, consists of images harvested from Google Street View annotated with word-level axis-aligned bounding boxes. *SVT* is more challenging than the *ICDAR* data as it contains smaller and lower resolution text. Furthermore, not all instances of text are annotated. In practice, this means that precision is heavily underestimated in evaluation. Lexicons consisting of 50 distractor words along with the ground-truth words are provided for each image; we refer to testing on *SVT* with these lexicons as *SVT-50*.

4.2. Text Localisation Experiments

We evaluate our detection networks to — (1) compare the performance when applied to single-scale and multiple down-scaled versions of the image and, (2) improve upon the state-of-the-art results in text detection when used as high-quality proposals.

Training. FCRN is trained on 800,000 images from our *SynthText in the Wild* dataset. Each image is resized to a size of 512×512 pixels. We optimise using SGD with momentum and batch-normalisation [18] after every convolutional layer (except the last one). We use mini-batches of 16 images each, set the momentum to 0.9, and use a weight-decay of 5^{-4}. The learning rate is set to 10^{-4} initially and is reduced to 10^{-5} when the training loss plateaus.

As only a small number (1-2%) of grid-cells contain text, we weigh down the non-text probability error terms initially by multiplying with 0.01; this weight is gradually increased to 1 as the training progresses. Due to class imbalance, all the probability scores collapse to zero if such a weighting scheme is not used.

Inference. We get the class probabilities and bounding-box predictions from our FCRN model. The predictions are filtered by thresholding the class probabilities (at a threshold t). Finally, multiple detections from nearby cells are suppressed using non-maximal suppression, whereby amongst two overlapping detections the one with the lower probability is suppressed. In the following we first give results for a conservative threshold of $t = 0.3$, for higher precision, and then relax this to $t = 0.0$ (i.e., all proposals accepted) for higher recall.

Evaluation protocol. We report text detection performance using two protocols commonly used in the literature — (1) *DetEval* [45] popularly used in ICDAR competitions for evaluating localisation methods, and (2) PASCAL VOC style intersection-over-union overlap method (≥ 0.5 IoU for a positive detection).

Single & multi-scale detection. The "FCRN single-scale"

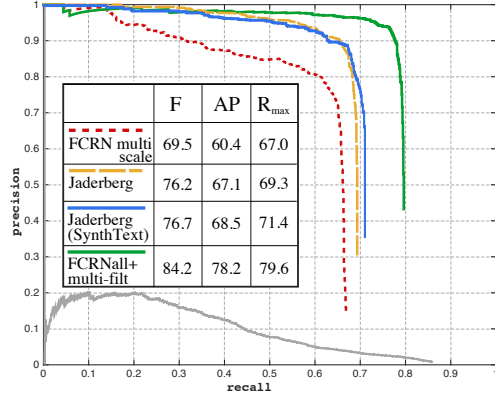|	F	AP	R_{max}
FCRN multi scale	69.5	60.4	67.0
Jaderberg	76.2	67.1	69.3
Jaderberg (SynthText)	76.7	68.5	71.4
FCRNall+ multi-filt	84.2	78.2	79.6

Figure 5. Precision-Recall curves for various text detection methods on IC13. The methods are: (1) multi-scale application of FCRN ("FCRN-multi"); (2) The original curve of Jaderberg *et al*. [20]; (3) Jaderberg *et al*. [20] retrained on the *SynthText in the Wild* dataset; and, (4) "FCRNall + multi-filt" methods. Maximum F-score (F), Average Precision (AP) and maximum Recall (R_{max}) are also given. The gray curve at the bottom is of multi-scale detections from our FCRN network (max. recall = 85.9%), which is fed into the multi-filtering post-processing to get the refined "FCRNall + multi-filt" detections.

entry in Table 2 shows the performance of our FCRN model on the test datasets. The precision at maximum F-measure of single-scale FCRN is comparable to the methods of Neuman *et al*. [33, 34], while the recall is significantly worse by 12%.

The "FCRN multi-scale" entry in Table 2 shows performance on multi-scale application of our network. This method improves maximum recall by more than 12% over the single-scale method and outperforms the methods of Neumann *et al*.

Post-processing proposals. Current end-to-end text spotting (detection and recognition) methods [1, 20, 44] boost performance by combining detection with text recognition. To further improve FCRN detections, we use the multi-scale detections from FCRN as proposals and refine them by using the post-processing stages of Jaderberg *et al*. [20]. There are three stages: first filtering using a binary text/no-text random-forest classifier; second, regressing an improved bounding-box using a CNN; and third recognition based NMS where the word images are recognised using a large fixed lexicon based CNN, and the detections are merged through non-maximal suppression based on word identities. Details are given in [20]. We use code provided by the authors for fair comparison.

We test this in two modes — (1) *low-recall*: where only high-scoring (probability > 0.3) multi-scale FCRN detections are used (the threshold previously used in the single- and multi-scale inference). This typically yields less than 30 proposals. And, (2) *high-recall*: where *all* the multi-scale FCRN detections (typically about a thousand in number) are used. Performance of these methods on text detec-
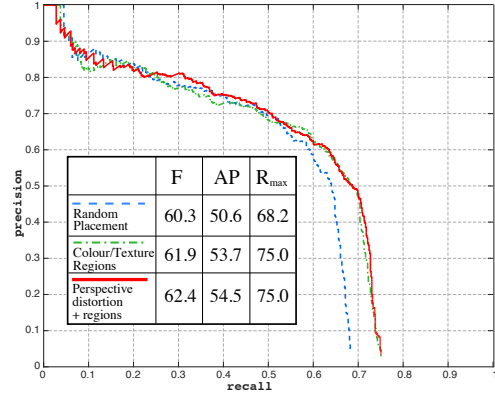

	F	AP	R_{max}
Random Placement	60.3	50.6	68.2
Colour/Texture Regions	61.9	53.7	75.0
Perspective distortion + regions	62.4	54.5	75.0

Figure 6. Precision-Recall curves text localisation on the SVT dataset using the model "FCRNall+multi-filt" when trained on increasingly sophisticated training sets (section 4.3).

tion are shown by the entries named "FCRN + multi-filt" and "FCRNall + multi-filt" respectively in Table 2. Note that the *low-recall* method achieves better than the state-of-the-art performance on text detection, whereas *high-recall* method significantly improves the state-of-the-art with an improvement of 6% in the F-measure for all the datasets.

Figure 5 shows the Precision-Recall curves for text detection on the *IC13* dataset. Note the high recall (85.9%) of the multi-scale detections output from FCRN before refinement using the multi-filtering post-processing. Also, note the drastic increase in maximum recall ($+10.3\%$) and in Average Precision ($+11.1\%$) for "FCRNall + multi-filt" as compared to Jaderberg *et al*.

Further, to establish that the improvement in text detection is due to the new detection model, and not merely due to the large size of our synthetic dataset, we trained Jaderberg *et al*.'s method on our *SynthText in the Wild* dataset – in particular, the ACF component of their region proposal stage.[4] Figure 5 and Table 2 show that, even with $10\times$ more (synthetic) training data, Jaderberg *et al*.'s model improves only marginally ($+0.8\%$ in AP, $+2.1\%$ in maximum recall).

A common failure mode is text in unusual fonts which are not present in the training set. The detector is also confused by symbols or patterns of constant stroke width which look like text, for example road-signs, stick figures etc. Since the detector does not scale the image up, extremely small sized text instances are not detected. Finally, words get broken into multiple instances or merged into one instance due to large or small spacing between the characters.

4.3. Synthetic Dataset Evaluation

We investigate the contribution that the various stages of the synthetic text-scene data generation pipeline bring to

---

[4]Their other region proposal method, EdgeBoxes, was not re-trained; as it is learnt from low-level edge features from the Berkeley Segmentation Dataset, which is not text specific.

Model	IC11	IC11*	IC13	SVT	SVT-50
Wang [42]	-	-	-	-	38
Wang & Wu [44]	-	-	-	-	46
Alsharif [1]	-	-	-	-	48
Neumann [34]	-	45.2	-	-	-
Jaderberg [21]	-	-	-	-	56
Jaderberg [20]	76	69	76	53	**76**
FCRN + multi-filt	80.5 (77.8)	75.8 (73.5)	80.3 (77.8)	54.7	68.0
FCRNall + multi-filt	**84.3** (81.2)	**81.0** (78.4)	**84.7** (81.8)	**55.7**	67.7

Table 3. Comparison with previous methods on end-to-end text spotting. Maximum F-measure% is reported. IC11* is evaluated according to the protocol described in [34]. Numbers in parenthesis are obtained if words containing non-alphanumeric characters are not ignored – SVT does not have any of these.

localisation accuracy: We generate three synthetic training datasets with increasing levels of sophistication, where the text is (1) is placed at random positions within the image, (2) restricted to the local colour and texture boundaries, and (3) distorted perspectively to match the local scene depth (while also respecting the local colour and texture boundaries as in (2) above). All other aspects of the datasets were kept the same — e.g. the text lexicon, background images, colour distribution.

Figure 6 shows the results on localisation on the *SVT* dataset of our method "FCRNall+multi-filt". Compared to random placement, restricting text to the local colour and texture regions significantly increases the maximum recall (+6.8%), AP (+3.85%), and the maximum F-measure (+2.1%). Marginal improvements are seen with the addition of perspective distortion: +0.75% in AP, +0.55% in maximum F-measure, and no change in the maximum recall. This is likely due to the fact that most text instances in the *SVT* datasets are in a fronto-parallel orientation. Similar trends are observed with the *ICDAR* 2013 dataset, but with more contained differences probably due to the fact that *ICDAR*'s text instances are much simpler than *SVT*'s and benefit less from the more advanced datasets.

4.4. End-to-End Text Spotting

Text spotting is limited by the detection stage, as state-of-the-art cropped word image recognition accuracy is over 98% [19]. We utilise our improvements in text localisation to obtain state-of-the-art results in text spotting.

Evaluation protocol. Unless otherwise stated, we follow the standard evaluation protocol by Wang *et al.* [42], where all words that are either less than three characters long or contain non-alphanumeric characters are ignored. An overlap (IoU) of at least 0.5 is required for a positive detection.

Table 3 shows the results on end-to-end text spotting task using the "FCRN + multi-filt" and "FCRNall + multi-filt" methods. For recognition we use the output of the interme-

	Total Time	Region Proposal	Proposal Filtering	BB-regression & recognition
FCRN+multi-filt	0.30	0.07	0.03	0.20
FCRNall+multi-filt	2.47	0.07	1.20	1.20
Jaderberg *et al.*	7.00	3.00	3.00	1.00

Table 4. Comparison of end-to-end text-spotting time (in seconds).

diary recognition stage of the pipeline based on the lexicon-encoding CNN of Jaderberg *et al.* [19]. We improve upon previously reported results (F-measure): +8% on the *IC-DAR* datasets, and +3% on the *SVT* dataset. Given the high recall of our method (as noted before in Figure 5), the fact that many text instances are unlabelled in *SVT* cause precision to drop; hence, we see smaller gains in *SVT* and do worse on *SVT-50*.

4.5. Timings

At test time FCRN can process 20 images per second (of size $512{\times}512$px) at single scale and about 15 images per second when run on multiple scales (1,1/2,1/4,1/8) on a GPU. When used as high-quality proposals in the text localisation pipeline of Jaderberg *et al.* [20], it replaces the region proposal stage which typically takes about 3 seconds per image. Hence, we gain a speed-up of about 45 times in the region proposal stage. Further, the "FCRN + multi-filt" method, which uses only the high-scoring detections from multi-scale FCRN and achieves state-of-the-art results in detection and end-to-end text spotting, cuts down the number of proposals in the later stages of the pipeline by a factor of 10: the region proposal stage of Jaderberg *et al.* proposes about 2000 boxes which are quickly filtered using a random-forest classifier to a manageable set of about 200 proposals, whereas the high-scoring detections from multi-scale FCRN are typically less than 30. Table 4 compares the time taken for end-to-end text-spotting; our method is between $3\times$ to $23\times$ faster than Jaderberg *et al.*'s, depending on the variant.

5. Conclusion

We have developed a new CNN architecture for generating text proposals in images. It would not have been possible to train this architecture on the available annotated datasets, as they contain far too few samples, but we have shown that training images of sufficient verisimilitude can be generated synthetically, and that the CNN trained *only* on these images exceeds the state-of-the-art performance for both detection and end-to-end text spotting on real images.

References

[1] O. Alsharif and J. Pineau. End-to-end text recognition with hybrid HMM maxout models. *ArXiv e-prints*, Oct 2013.

[2] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *IEEE PAMI*, 33:898–916, 2011.

[3] P. Arbelaez, J. Pont-Tuset, J. Barron, F. Marques, and J. Malik. Multiscale combinatorial grouping. In *Proc. CVPR*, 2014.

[4] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In *Proc. ECCV*, 2014.

[5] A. Criminisi, I. D. Reid, and A. Zisserman. Single view metrology. In *Proc. ICCV*, pages 434–442, 1999.

[6] N. Dalal and B. Triggs. Histogram of Oriented Gradients for Human Detection. In *Proc. CVPR*, volume 2, pages 886–893, 2005.

[7] P. Dollar, R. Appel, and S. Belongie. Fast feature pyramids for object detection. *IEEE PAMI*, 36(8):1532–1545, 2014.

[8] A. Dosovitskiy and T. Brox. Inverting visual representations with convolutional networks. In *Proc. CVPR*, 2016. To appear.

[9] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. Smagt, D. Cremers, and T. Brox. Flownet: Learning optical flow with convolutional networks. In *Proc. ICCV*, 2015.

[10] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Comm. ACM*, 24(6):381–395, 1981.

[11] R. B. Girshick. Fast R-CNN. In *Proc. ICCV*, 2015.

[12] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proc. CVPR*, 2014.

[13] I. J. Goodfellow, Y. Bulatov, J. Ibarz, S. Arnoud, and V. Shet. Multi-digit number recognition from street view imagery using deep convolutional neural networks. In *Proc. ICLR*, 2014.

[14] D. Hoiem, A. A. Efros, and M. Hebert. Automatic photo pop-up. In *Proc. ACM SIGGRAPH*, 2005.

[15] D. Hoiem, A. A. Efros, and M. Hebert. Geometric context from a single image. In *Proc. ICCV*, 2005.

[16] P. V. C. Hough. Method and means for recognizing complex patterns. US Patent 3,069,654, 1962.

[17] W. Huang, Y. Qiao, and X. Tang. Robust scene text detection with convolution neural network induced mser trees. In *Proc. ECCV*, 2014.

[18] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proc. ICML*, 2015.

[19] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman. Synthetic data and artificial neural networks for natural scene text recognition. In *Workshop on Deep Learning, NIPS*, 2014.

[20] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman. Reading text in the wild with convolutional neural networks. *IJCV*, 2015.

[21] M. Jaderberg, A. Vedaldi, and A. Zisserman. Deep features for text spotting. In *Proc. ECCV*, 2014.

[22] A. Janoch, S. Karayev, Y. Jia, J. T. Barron, M. Fritz, K. Saenko, and T. Darrell. A category-level 3-d object dataset: Putting the kinect to work. In *ICCV Workshop on Consumer Depth Cameras in Computer Vision*, 2011.

[23] D. Karatzas, L. Gomez-Bigorda, A. Nicolaou, S. Ghosh, A. Bagdanov, M. Iwamura, J. Matas, L. Neumann, V. R. Chandrasekhar, S. Lu, et al. ICDAR 2015 robust reading competition. In *Proc. ICDAR*, pages 1156–1160, 2015.

[24] D. Karatzas, F. Shafait, S. Uchida, M. Iwamura, S. R. Mestre, J. Mas, D. F. Mota, J. A. Almazan, L. P. de las Heras, et al. ICDAR 2013 robust reading competition. In *Proc. ICDAR*, pages 1484–1493, 2013.

[25] K. Karsch, V. Hedau, D. Forsyth, and D. Hoiem. Rendering synthetic objects into legacy photographs. *ACM Transactions on Graphics*, 30(6):157, 2011.

[26] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *NIPS*, pages 1106–1114, 2012.

[27] K. Lang and T. Mitchell. Newsgroup 20 dataset, 1999.

[28] B. Leibe, A. Leonardis, and B. Schiele. Combined object categorization and segmentation with an implicit shape model. In *Workshop on Statistical Learning in Computer Vision, ECCV*, May 2004.

[29] K. Lenc and A. Vedaldi. R-CNN minus R. In *Proc. BMVC.*, 2015.

[30] F. Liu, C. Shen, and G. Lin. Deep convolutional neural fields for depth estimation from a single image. In *Proc. CVPR*, 2015.

[31] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proc. CVPR*, 2015.

[32] A. Mishra, K. Alahari, and C. Jawahar. Scene text recognition using higher order language priors. *Proc. BMVC.*, 2012.

[33] L. Neumann and J. Matas. Real-time scene text localization and recognition. In *Proc. CVPR*, volume 3, pages 1187–1190, 2012.

[34] L. Neumann and J. Matas. Scene text localization and recognition with oriented stroke detection. In *Proc. ICCV*, pages 97–104, December 2013.

[35] P. Perez, M. Gangnet, and A. Blake. Poisson image editing. *ACM Transactions on Graphics*, 22(3):313–318, 2003.

[36] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *Proc. CVPR*, 2016. To appear.

[37] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NIPS*, 2016.

[38] A. Saxena, M. Sun, and A. Y. Ng. Make3d: Learning 3d scene structure from a single still image. *IEEE PAMI*, 31(5):824–840, 2009.

[39] A. Shahab, F. Shafait, and A. Dengel. ICDAR 2011 robust reading competition challenge 2: Reading text in scene images. In *Proc. ICDAR*, pages 1491–1496, 2011.

[40] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from rgbd images. In *Proc. ECCV*, 2012.

[41] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

[42] K. Wang, B. Babenko, and S. Belongie. End-to-end scene text recognition. In *Proc. ICCV*, pages 1457–1464, 2011.

[43] K. Wang and S. Belongie. Word spotting in the wild. In *Proc. ECCV*, 2010.

[44] T. Wang, D. J. Wu, A. Coates, and A. Y. Ng. End-to-end text recognition with convolutional neural networks. In *Proc. ICPR*, pages 3304–3308, 2012.

[45] 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.

[46] I. Yildirim, T. D. Kulkarni, W. A. Freiwald, and J. B. Tenenbaum. Efficient and robust analysis-by-synthesis in vision: A computational framework, behavioral tests, and modeling neuronal representations. In *Annual Conference of the Cognitive Science Society*, 2015.

[47] Z. Zhang, W. Shen, C. Yao, and X. Bai. Symmetry-based text line detection in natural scenes. In *Proc. CVPR*, 2015.

[48] C. L. Zitnick and P. Dollar. Edge boxes: Locating object proposals from edges. In *Proc. ECCV*, pages 391–405, 2014.