Multi-Oriented Scene Text Detection via Corner Localization and Region Segmentation

Pengyuan Lyu[1], Cong Yao[2], Wenhao Wu[2], Shuicheng Yan[3], Xiang Bai[1]
[1]Huazhong University of Science and Technology
[2]Megvii Technology Inc.
[3]National University of Singapore
{lvpyuan, yaocong2010}@gmail.com, wwh@megvii.com, eleyans@nus.edu.sg, xbai@hust.edu.cn

Abstract

Previous deep learning based state-of-the-art scene text detection methods can be roughly classified into two categories. The first category treats scene text as a type of general objects and follows general object detection paradigm to localize scene text by regressing the text box locations, but troubled by the arbitrary-orientation and large aspect ratios of scene text. The second one segments text regions directly, but mostly needs complex post processing. In this paper, we present a method that combines the ideas of the two types of methods while avoiding their shortcomings. We propose to detect scene text by localizing corner points of text bounding boxes and segmenting text regions in relative positions. In inference stage, candidate boxes are generated by sampling and grouping corner points, which are further scored by segmentation maps and suppressed by NMS. Compared with previous methods, our method can handle long oriented text naturally and doesn't need complex post processing. The experiments on ICDAR2013, ICDAR2015, MSRA-TD500, MLT and COCO-Text demonstrate that the proposed algorithm achieves better or comparable results in both accuracy and efficiency. Based on VGG16, it achieves an F-measure of 84.3% *on ICDAR2015 and* 81.5% *on MSRA-TD500.*

1. Introduction

Recently, extracting textual information from natural scene images has become increasingly popular, due to the growing demands of real-world applications (e.g., product search [4], image retrieval [19], and autonomous driving). Scene text detection, which aims at locating text in natural images, plays an important role in various text reading systems [34, 10, 47, 5, 20, 13, 7, 25].

Scene text detection is challenging due to both external and internal factors. The external factors come from the environment, such as noise, blur and occlusion, which are also major problems disturbing general object detection. The internal factors are caused by properties and variations of scene text. Compared with general object detection, scene text detection is more complicated because: 1) Scene text may exist in natural images with arbitrary orientation, so the bounding boxes can also be rotated rectangles or quadrangles; 2) The aspect ratios of bounding boxes of scene text vary significantly; 3) Since scene text can be in the form of characters, words, or text lines, algorithms might be confused when locating the boundaries.

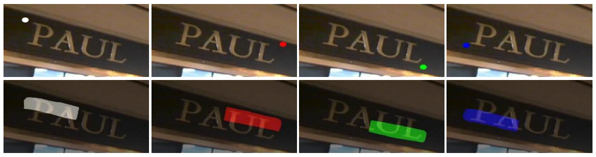


Figure 1. The images in top row and bottom row are the predicted corner points and position-sensitive maps in top-left, top-right, bottom-right, bottom-left order, respectively.

In the past few years, scene text detection has been widely studied [10, 5, 49, 20, 43, 52, 39, 42] and has achieved obvious progresses recently, with the rapid development of general object detection and semantic segmentation. Based on general object detection and semantic segmentation models, several well-designed modifications are made to detect text more accurately. Those scene text detectors can be split into two branches. The first branch is based on general object detectors (SSD [30], YOLO [37] and DenseBox [18]), such as TextBoxes [27], FCRN [14] and EAST [53] *etc.*, which predict candidate bounding boxes directly. The second branch is based on semantic segmentation, such as [52] and [50], which generate segmentation maps and produce the final text bounding boxes by post-processing.

Different from previous methods, in this paper we combine the ideas of object detection and semantic segmentation and apply them in an alternative way. Our motivations mainly come from two observations: 1) a rectangle can be

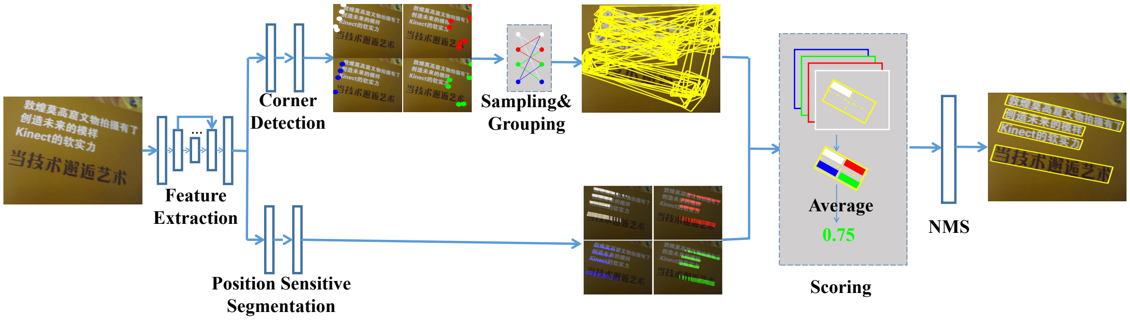


Figure 2. Overview of our method. Given an image, the network outputs corner points and segmentation maps by corner detection and position-sensitive segmentation. Then candidate boxes are generated by sampling and grouping corner points. Finally, those candidate boxes are scored by segmentation maps and suppressed by NMS.

determined by corner points, regardless of the size, aspect ratio or orientation of the rectangle; 2) region segmentation maps can provide effective location information of text. Thus, we first detect the corner points (top-left, top-right, bottom-right, bottom-left, as shown in Fig. 1) of text region rather than text boxes directly. Besides, we predict position-sensitive segmentation maps (shown in Fig. 1) instead of a text/non-text map as in [52] and [50]. Finally, we generate candidate bounding boxes by sampling and grouping the detected corner points and then eliminate unreasonable boxes by segmentation information. The pipeline of our proposed method is depicted in Fig. 2.

The key advantages of the proposed method are as follows: 1) Since we detect scene text by sampling and grouping corner points, our approach can naturally handle arbitrary-oriented text; 2) As we detect corner points rather than text bounding boxes, our method can spontaneously avoid the problem of large variation in aspect ratio; 3) With position-sensitive segmentation, it can segment text instances well, no matter the instances are characters, words, or text lines; 4) In our method, the boundaries of candidate boxes are determined by corner points. Compared with regressing text bounding box from anchors ([27, 32]) or from text regions ([53, 16]), the yielded bounding boxes are more accurate, particularly for long text.

We validate the effectiveness of our method on horizontal, oriented, long and oriented text as well as multi-lingual text from public benchmarks. The results show the advantages of the proposed algorithm in accuracy and speed. Specifically, the F-Measures of our method on ICDAR2015 [22], MSRA-TD500 [49] and MLT [2] are 84.3%, 81.5% and 72.4% respectively, which outperform previous state-of-the-art methods significantly. Besides, our method is also competitive in efficiency. It can process more than **10.4** images (512x512 in size) per second.

The contributions of this paper are four-fold: (1) We propose a new scene text detector that combines the ideas of object detection and segmentation, which can be trained and evaluated end-to-end. (2) Based on position-sensitive ROI pooling [9], we propose a rotated position-sensitive ROI average pooling layer that can handle arbitrary-oriented proposals. (3) Our method can simultaneously handle the challenges (such as rotation, varying aspect ratios, very close instances) in multi-oriented scene text, which are suffered by previous methods. (4) Our method achieves better or competitive results in both accuracy and efficiency.

2. Related Work

2.1. Regression Based Text Detection

Regression based text detection has become the mainstream of scene text detection in the past two years. Based on general object detectors, several text detection methods were proposed and achieved substantial progress. Originating from SSD [30], TextBoxes [27] use ”long” default boxes and ”long” convolutional filters to cope with the extreme aspect ratios. Similarly, in [32] Ma *et al.* utilize the architecture of Faster-RCNN [38] and add rotated anchors in RPN to detect arbitrary-oriented scene text. SegLink [39] predicts text segments and the linkage of them in a SSD style network and links the segments to text boxes, in order to handle long oriented text in natural scene. Based on DenseBox [18], EAST [53] regresses text boxes directly.

Our method is also adapted from a general object detector DSSD [11]. But unlike the above methods that regress text boxes or segments directly, we propose to localize the positions of corner points, and then generate text boxes by sampling and grouping the detected corners.

2.2. Segmentation Based Text Detection

Segmentation based text detection is another direction of text detection. Inspired by FCN [31], some methods are proposed to detect scene text by using segmentation maps. In [52], Zhang *et al.* first attempt to extract text blocks from a segmentation map by a FCN. Then they detect characters in those text blocks with MSER [34] and group the characters to words or text lines by some priori rules. In [50], Yao *et al.* use a FCN to predict three types of maps (text regions, characters, and linking orientations) of the input images. Then some post-processings are conducted to obtain text bounding boxes with the segmentation maps.

Different from the previous segmentation based text detection methods, which usually need complex post-processing, our method is simpler and clearer. In inference stage, the position-sensitive segmentation maps are used to score the candidate boxes by our proposed **Rotated Position-Sensitive Average ROI Pooling** layer.

2.3. Corner Point Based General Object Detection

Corner point based general object detection is a new stream of general object detection methods. In DeNet [45], Tychsen-Smith *et al.* propose a corner detect layer and a sparse sample layer to replace RPN in a Faster-RCNN style two-stage model. In [48], Wang *et al.* propose PLN (Point Linking Network) which regresses the corner/center points of bounding-box and their links using a fully convolutional network. Then the bounding boxes of objects are formed using the corner/center points and their links.

Our method is inspired by those corner point based object detection methods, but there are key differences. First, the corner detector of our method is different. Second, we use segmentation map to score candidate boxes. Third, it can produce arbitrary-oriented boxes for objects (text).

2.4. Position-Sensitive Segmentation

Recently, instance-aware semantic segmentation methods are proposed with position-sensitive maps. In [8], Dai *et al.* first introduce relative position to segmentation and propose InstanceFCN for instance segment proposal. In FCIS [26], with the assistance of position-sensitive inside/outside score maps, Li *et al.* propose an end-to-end network for instance-aware semantic segmentation.

We also adopt position-sensitive segmentation maps to predict text regions. Compared with the above-mentioned methods, there are three key differences: 1) We optimize the network with position-sensitive ground truth directly (detailed in Sec 4.1.1); 2) Our position-sensitive maps can be used to predict text regions and score proposals simultaneously (detailed in Sec 4.2.2), different from FCIS which uses two types of position-sensitive maps (inside and outside); 3) Our proposed Rotated Position-Sensitive ROI Average Pooling can handle arbitrary-oriented proposals.

3. Network

The network of our method is a fully convolutional network that plays the roles of feature extraction, corner detection and position-sensitive segmentation. The network architecture is shown in Fig. 3. Given an image, the network produces candidate corner points and segmentation maps.

3.1. Feature Extraction

The backbone of our model is adapted from a pre-trained VGG16 [41] network and designed with the following considerations: 1) the size of scene text varies hugely, so the backbone must has enough capacity to handle this problem well; 2) backgrounds in natural scenes are complex, so the features should better contain more context. Inspired by the good performance achieved on those problem by FPN [28] and DSSD [11], we adopt the backbone in FPN/DSSD architecture to extract features.

In detail, we convert the fc6 and fc7 in the VGG16 to convolutional layers and name them conv6 and conv7 respectively. Then several extra convolutional layers (conv8, conv9, conv10, conv11) are stacked above conv7 to enlarge the receptive fields of extracted features. After that, a few deconvolution modules proposed in DSSD [11] are used in a top-down pathway (Fig. 3). Particularly, to detect text with different sizes well, we cascade deconvolution modules with 256 channels from conv11 to conv3 (the features from conv10, conv9, conv8, conv7, conv4, conv3 are reused), and 6 deconvolution modules are built in total. Including the features of conv11, we name those output features $F_3, F_4, F_7, F_8, F_9, F_{10}$ and F_{11} for convenience. In the end, the feature extracted by conv11 and deconvolution modules which have richer feature representations are used to detect corner points and predict position-sensitive maps.

3.2. Corner Detection

For a given rotated rectangular bounding box $R = (x, y, w, h, \theta)$, there are 4 corner points (top-left, top-right, bottom-right, bottom-left) and can be represented as two-dimensional coordinates $\{(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4)\}$ in a clockwise direction. To expediently detect corner points, here we redefine and represent a corner point by a horizontal square $C = (x_c, y_c, ss, ss)$, where x_c, y_c are the coordinate of a corner point (such as x_1, y_1 for top-left point) as well as the center of the horizontal square. ss is the length short side of the rotated rectangular bounding box R.

Following SSD and DSSD, we detect corner points with default boxes. Different from the manner in SSD or DSSD where each default box outputs the classification scores and offsets of the corresponding candidate box, corner point detection is more complex because there might be more than one corner points in the same location (such as a location can be the bottom-left corner and top-right corner of two boxes simultaneously). So in our case, a default box should output classification scores and offsets for 4 candidate boxes corresponding to the 4 types of corner points.

We adapt the prediction module proposed in [11] to predict scores and offsets in two branches in a convolutional manner. In order to reduce the computational complexity, the filters of all convolutions are set to 256. For an $m \times n$

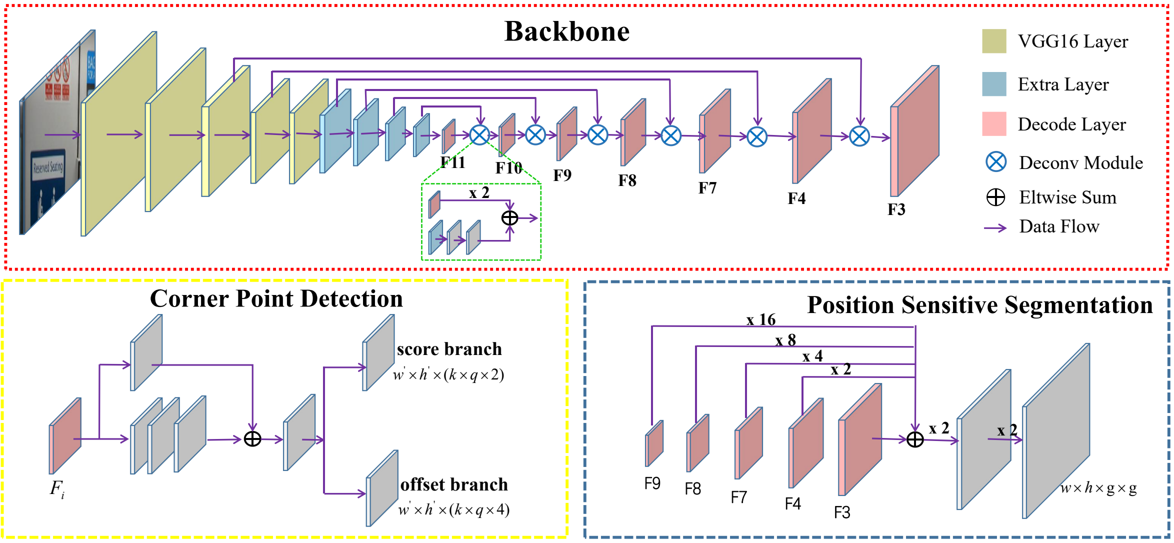


Figure 3. Network Architecture. The network contains three parts: backbone, conner point detector and position-sensitive segmentation predictor. The backbone is adapted from DSSD [11]. Conner point detectors are built on multiple feature layers (blocks in pink). position-sensitive segmentation predictor shares some features (pink blocks) with corner point detectors.

feature map with k default boxes in each cell, the ”score” branch and ”offset” branch output 2 scores and 4 offsets respectively for each type of corner point of each default box. Here, 2 for ”score” branch means whether a corner point exists in this position. In total, the output channels of the ”score” branch and the ”offset” branch are $k \times q \times 2$ and $k \times q \times 4$, where q means the type of corner points. By default, q is equal to 4.

In the training stage, we follow the matching strategy of default boxes and ground truth ones in SSD. To detect scene text with different sizes, we use default boxes of multiple sizes on multiple layer features. The scales of all default boxes are listed in Table 1. The aspect ratios of default boxes are set to 1.

3.3. Position-Sensitive Segmentation

In the previous segmentation based text detection methods [52, 50], a segmentation map is generated to represent the probability of each pixel belonging to text regions. However those text regions in score map always can not be separated from each other, as a result of the overlapping of text regions and inaccurate predictions of text pixels. To get the text bounding boxes from the segmentation map, complex post-processing are conducted in [52, 50].

Inspired by InstanceFCN [8], we use position-sensitive segmentation to generate text segmentation maps. Compared with previous text segmentation methods, relative positions are generated. In detail, for a text bounding box R, a $g \times g$ regular grid is used to divide the text bounding box into multiple bins (*i.e.*, for a 2×2 grid, a text region can be split into 4 bins, that is top-left, top-right, bottom-right, bottom-left). For each bin, a segmentation map is used to determine whether the pixels in this map belong to this bin.

We build position-sensitive segmentation with corner point detection in a unified network. We reuse the features of F_3, F_4, F_7, F_8, F_9 and build some convolutional blocks on them follow the residual block architecture of corner point detection branch (Shown in Fig. 3). All outputs of those blocks are resized to the scale of F_3 by bilinear upsampling with the scale factors set to 1, 2, 4, 8, 16. Then all those outputs with the same scale are added together to generate richer features. We further enlarge the resolution of fused features by two continuous *Conv1x1-BN-ReLU-Deconv2x2* blocks and set the kernels of the last deconvolution layer to $g \times g$. So, the final position-sensitive segmentation maps have $g \times g$ channels and the same size as the input images. In this work, we set g to 2 in default.

4. Training and Inference

4.1. Training

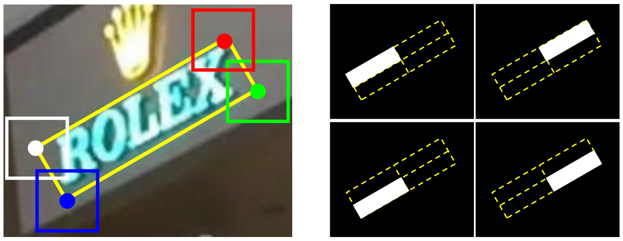
4.1.1 Label Generation

For an input training sample, we first convert each text box in ground truth into a rectangle that covers the text box region with minimal area and then determine the relative position of 4 corner points.

We determine the relative position of a rotated rectangle by the following rules: 1) the x-coordinates of top-left and bottom-left corner points must less than the x-coordinates of top-right and bottom-right corner points; 2) the y-coordinates of top-left and top-right corner points must less than the y-coordinates of bottom-left and bottom-right corner points. After that, the original ground truth can be represented as a rotated rectangle with relative position

layer	F_3	F_4	F_7	F_8	F_9	F_{10}	F_{11}
scales	$4, 8, 6, 10, 12, 16$	$20, 24, 28, 32$	$36, 40, 44, 48$	$56, 64, 72, 80$	$88, 96, 104, 112$	$124, 136, 148, 160$	$184, 208, 232, 256$

Table 1. Scales of default boxes on different layers.



(a) (b)

Figure 4. Label generation for corner points detection and position-sensitive segmentation. (a) Corner points are redefined and represented by squares (boxes in white, red, green, blue) with the side length set as the short side of text bounding box R (yellow box). (b) Corresponding ground truth of R in (a) for position-sensitive segmentation.

of corner points. For convenience, we term the rotated rectangle $R = \{P_i | i \in \{1, 2, 3, 4\}\}$, where $P_i = (x_i, y_i)$ are the corner points of the rotated rectangle in top-left, top-right, bottom-right, bottom-left order.

We generate the label of corner point detection and position-sensitive segmentation using R. For corner point detection, we first compute the short side of R and represent the 4 corner points by horizontal squares as shown in Fig. 5 (a). For position-sensitive segmentation, we generate pixel-wise masks of text/non-text with R. We first initialize 4 masks with the same scale as the input image and set all pixel value to 0. Then we divide R into four bins with a 2×2 regular grid and assign each bin to a mask, such as top-left bin to the first mask. After that, we set the value of all pixels in those bins to 1, as shown in Fig. 5 (b).

4.1.2 Optimization

We train the corner detection and position-sensitive segmentation simultaneously. The loss function is defined as:

$$L = \frac{1}{N_c} L_{conf} + \frac{\lambda_1}{N_c} L_{loc} + \frac{\lambda_2}{N_s} L_{seg} \tag{1}$$

Where L_{conf} and L_{loc} are the loss functions of the score branch for predicting confidence score and the offset branch for localization in the module of corner point detection. L_{seg} is the loss function of position-sensitive segmentation. N_c is the number of positive default boxes, N_s is the number of pixels in segmentation maps. N_c and N_s are used to normalize the losses of corner point detection and segmentation. λ_1 and λ_2 are the balancing factors of the three tasks. In default, we set the λ_1 to 1 and λ_2 to 10.

We follow the matching strategy of SSD and train the score branch using Cross Entropy loss:

$$L_{conf} = CrossEntropy(y_c, p_c) \tag{2}$$

Where y_c is the ground truth of all default boxes, 1 for positive and 0 otherwise. p_c is the predicted scores. In consideration of the extreme imbalance between positive and negative samples, the category homogenization is necessary. We use the online hard negative mining proposed in [40] to balance training samples and set the ratio of positives to negatives to $1 : 3$.

For the offset branch, we regress the offsets relative to default boxes as Fast RCNN [12] and optimize them with Smooth L1 loss:

$$L_{loc} = SmoothL1(y_l, p_l) \tag{3}$$

Where $y_l = (\triangle x, \triangle y, \triangle s_s, \triangle s_s)$ is the ground truth of offset branch and $p_l = (\triangle\tilde{x}, \triangle\tilde{y}, \triangle\tilde{s_s}, \triangle\tilde{s_s})$ is the predicted offsets. The y_l can be calculated by a default box $B = (x_b, y_b, ss_b, ss_b)$ and a corner point box $C = (x_c, y_c, ss_c, ss_c)$:

$$\triangle x = \frac{x_b - x_c}{ss_b} \tag{4}$$

$$\triangle y = \frac{y_b - y_c}{ss_b} \tag{5}$$

$$\triangle ss = \log(\frac{ss_b}{ss_c}) \tag{6}$$

We train position-sensitive segmentation by minimizing the Dice loss [33]:

$$L_{seg} = 1 - \frac{2 y_s p_s}{y_s + p_s} \tag{7}$$

Where y_s is the label of position-sensitive segmentation and p_s is the prediction of our segmentation module.

4.2. Inference

4.2.1 Sampling and Grouping

In inference stage, many corner points are yielded with the predicted location, short side and confidence score. Points with high score (great than 0.5 in default) are kept. After NMS, 4 corner point sets are composed based on relative position information.

We generate the candidate bounding boxes by sampling and grouping the predicted corner points. In theory, a rotated rectangle can be constructed by two points and a side perpendicular to the line segment made up by the two points. For a predicted point, the short side is known, so

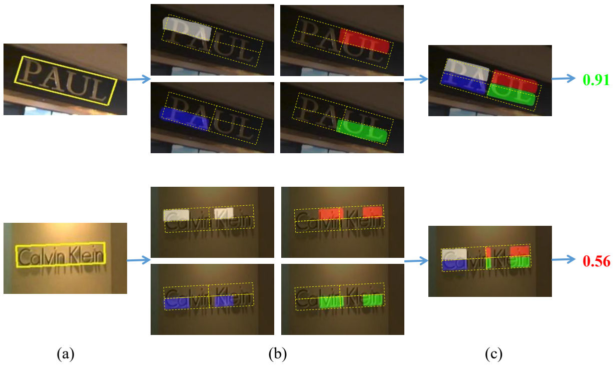


Figure 5. Overview of the scoring process. The yellow boxes in (a) are candidate boxes. (b) are predicted segmentation maps. We generate instance segment (c) of candidate boxes by assembling the segmentation maps as [8]. Scores are calculated by averaging the instance segment regions.

we can form a rotated rectangle by sampling and grouping two corner points in corner point sets arbitrarily, such as (top-left, top-right), (top-right, bottom-right), (bottom-left, bottom-right) and (top-left, bottom-left) pairs.

Several priori rules are used to filter unsuitable pairs: 1) the relative positional relations can not be violated, such as the x-coordinate of top-left point must less than that of top-right point in (top-left, top-right) pair; 2) the shortest side of the constructed rotated rectangle must be greater than a threshold (the default is 5); 3) the predicted short sides ss_1 and ss_2 of the two points in a pair must satisfy:

$$\frac{\max(ss_1, ss_2)}{\min(ss_1, ss_2)} \leq 1.5 \tag{8}$$

4.2.2 Scoring

A large number of candidate bounding boxes can be generated after sampling and grouping corner points. Inspired by InstanceFCN[8] and RFCN [9], we score the candidate bounding boxes by the position-sensitive segmentation maps. The processes are shown in Fig. 5.

To handle the rotated text bounding boxes, we adapt the Position-Sensitive ROI pooling layer in [9] and propose **Rotated Position-Sensitive ROI Average pooling layer**. Specifically, for a rotated box, we first split the box into $g \times g$ bins. Then we generate a rectangle for each bin with the minimum area to cover the bin. We loop over all pixels in the minimum rectangle and calculate mean value of all pixels which in the bin. In the end, the score of a rotated bounding box is obtained by averaging the means of $g \times g$ bins. The specific processes are shown in Algorithm 1.

The candidate boxes with low score will be filtered out. We set the threshold τ to 0.6 by default.

Algorithm 1 Rotated Position-Sensitive ROI Average Pooling

```
Input: rotated bounding box $B$, $g \times g$ regular grid $G$, Segmentation maps $S$
Generating $Bins$ by spitting $B$ with $G$.
$M \leftarrow 0, i \leftarrow 0$
for $i$ in $range(g \times g)$ do
    $bin \leftarrow Bins[i], C \leftarrow 0, P \leftarrow 0,$
    $R \leftarrow MiniRect(bin)$
    for $pixel$ in $R$ do
        if $pixel$ in $bin$ then
            $C \leftarrow C + 1, P \leftarrow P + G[i][pixel].value$
    $M \leftarrow M + \frac{P}{C}$
$score \leftarrow \frac{M}{g \times g}$
return score
```

5. Experiments

To validate the effectiveness of the proposed method, we conduct experiments on five public datasets: ICDAR2015, ICDAR2013, MSRA-TD500, MLT, COCO-Text, and compare with other state-of-the-art methods.

5.1. Datasets

SynthText [14] is a synthetically generated dataset which consists of about 800000 synthetic images. We use the dataset with word level labels to pre-train our model.

ICDAR2015 is a dataset proposed in the Challenge 4 of the 2015 Robust Reading Competition [22] for incidental scene text detection. There are 1000 images for training and 500 images for testing with annotations labeled as word level quadrangles.

ICDAR2013 is a dataset proposed in the Challenge 2 of the 2013 Robust Reading Competition [23] focuses on horizontal text in scene. It contains 229 images for training and 233 images for testing.

MSRA-TD500 [49] is a dataset collected for detecting arbitrary-oriented long text lines. It consists of 300 training images and 200 test images with text line level annotations.

MLT is a dataset that proposed on ICDAR2017 Competition [2] and focuses on the multi-oriented, multi-script and multi-lingual aspects of scene text. It consists of 7200 training images, 2000 validation images and 9000 test images.

COCO-Text [46] is a large scale scene text dataset which comes from the MS COCO dataset [29]. There are 63686 images are annotated and two versions of annotations and splits (V1.1 and V1.4) are released by the official. Previous methods are all evaluated on V1.1 and the new V1.4 are used on ICDAR2017 Competition [1].

5.2. Implementation Details

Training Our model is pre-trained on SynthText then finetuned on other datasets (except COCO-Text). We use Adam [24] to optimize our model with the learning rate fixed to $1e-4$. In pre-train stage, we train our model on SynthText for one epoch. During finetuning stage, the number of iterations are decided by the sizes of datasets.

Data Augmentation We use the same way of data augmentation as SSD. We randomly sample a patch from the input image in the manner of SSD, then resize the sampled patch to 512×512.

Post Processing NMS is the only post processing step of our method. We set the threshold of NMS to 0.3.

Our method is implemented in PyTorch [3]. All the experiments are conducted on a regular workstation (CPU: Intel(R) Xeon(R) CPU E5-2650 v3 @ 2.30GHz; GPU:Titan Pascal; RAM: 64GB). We train our model with the batch size of 24 on 4 GPUs in parallel and evaluate our model on 1 GPU with batch size set as 1.

5.3. Detecting Oriented Text

We evaluate our model on the ICDAR2015 dataset to test its ability of arbitrarily oriented text detection. We finetune our model another 500 epochs on the datasets of ICDAR2015 and ICDAR2013. Note that, to detect vertical text better, in the last 15 epochs, we randomly rotate the sampled patches by 90 degree or -90 degree with the probability of 0.2. In testing, we set τ to 0.7 and resize the input images to 768×1280. Following [53, 17, 16], we also evaluate our model on ICDAR2015 with multi-scale inputs, $\{512\times512, 768\times768, 768\times1280, 1280\times1280\}$ in default.

We compare our method with other state-of-the-art methods and list all the results in Table 2. Our method outperforms the previous methods by a large margin. When tested at single scale, our method achieves the F-measure of 80.7%, which surpasses all competitors [52, 44, 50, 39, 53, 15] . Our method achieves 84.3% in F-measure with multi-scale inputs, higher than the current best one [16] by 3.3%.

To explore the gain between our method which detects corner points and the method which regresses text boxes directly, we train a network named ”baseline” in Table. 2 using the same settings as our method. The baseline model consists of the same backbone as our method and the prediction module in SSD/DSSD. With slight time cost, our method boost the accuracy greatly (53.3% *vs* 80.7%).

5.4. Detecting Horizontal Text

We evaluate the ability of our model to detect horizontal text on ICDAR2013 dataset. We further train our model on ICDAR2013 dataset for 60 epochs on the basis of the finetuned ICDAR2015 model. In testing, the input images are resized to 512×512. We also use multi-scale inputs to evaluate our model.

Method	Precision	Recall	F-measure	FPS
Zhang *et al.* [52]	70.8	43.0	53.6	0.48
CTPN [44]	74.2	51.6	60.9	7.1
Yao *et al.* [50]	72.3	58.7	64.8	1.61
SegLink [39]	73.1	**76.8**	75.0	-
EAST [53]	80.5	72.8	76.4	6.52
SSTD [15]	80.0	73.0	77.0	**7.7**
baseline	66.0	44.7	53.3	4.5
ours	**94.1**	70.7	**80.7**	3.6
EAST *† [53]	83.3	78.3	80.7	-
WordSup * [17]	79.3	77.0	78.2	**2**
He *et al.* *† [16]	82.0	**80.0**	81.0	1.1
ours*	**89.5**	79.7	**84.3**	1

Table 2. Results on ICDAR2015. * means multi-scale, † stands for the base net of the model is not VGG16.

Method	Precision	Recall	F-measure	FPS
Neumann *et al.* [35]	81.8	72.4	77.1	3
Neumann *et al.* [36]	82.1	71.3	76.3	3
Fastext [6]	84.0	69.3	76.8	6
Zhang *et al.* [51]	88.0	74.0	80.0	0.02
Zhang *et al.* [52]	88.0	78.0	83.0	0.5
Yao *et al.* [50]	88.9	80.2	84.3	1.61
CTPN [44]	93.0	83.0	**88.0**	7.1
TextBoxes [27]	88.0	74.0	81.0	11
SegLink [39]	87.7	83.0	85.3	**20.6**
SSTD [15]	89.0	**86.0**	**88.0**	7.7
ours	**93.3**	79.4	85.8	10.4
FCRN * [14]	92.0	75.5	83.0	0.8
TextBoxes * [27]	89.0	83.0	86.0	1.3
He *et al.* *† [16]	92.0	81.0	86.0	1.1
WordSup * [17]	**93.3**	**87.5**	**90.3**	**2**
ours*	92.0	84.4	88.0	1

Table 3. Results on ICDAR2013. * means multi-scale, † stands for the base net of the model is not VGG16. Note that, the methods of the top three lines are evaluated under the ”ICDAR2013” evaluation protocol.

The results are listed in Table 3 and mostly are reported with the ”Deteval” evaluation protocol. Our method achieves very competitive results. When tested at single scale, our method achieves the F-measure of 85.8%, which is slightly lower than the highest result. Besides, our method can run at 10.4 FPS, faster than most methods. For multi-scale evaluation, our method achieves the F-measure of 88.0%, which is also competitive compared with other methods.

5.5. Detecting Long Oriented Text Line

On MSRA-TD500, we evaluate the performance of our method for detecting long and multi-lingual text lines. HUST-TR400 is also used as training data as the MSRA-TD500 only contains 300 training images. The model is initialized with the model pre-trained on SynthText and then

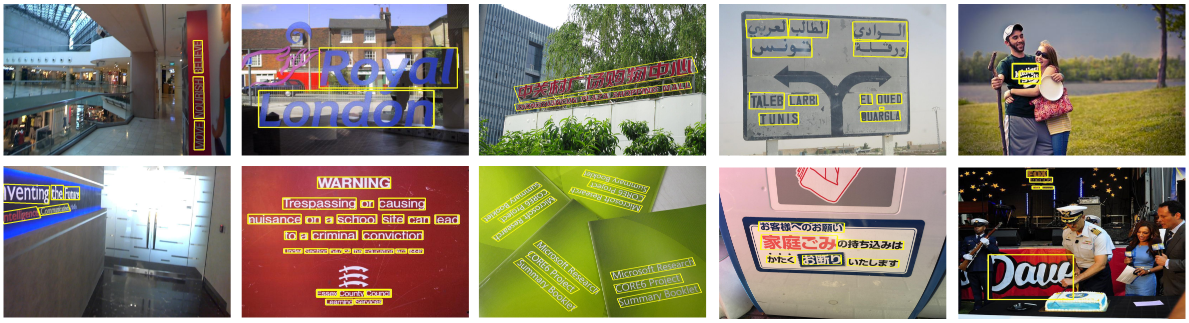


Figure 6. Examples of detection results. From left to right in columns: ICDAR2015, ICDAR2013, MSRA-TD500, MLT, COCO-Text.

Method	Precision	Recall	F-measure	FPS
TD-ICDAR [49]	53.0	52.0	50.0	-
TD-Mixture [49]	63.0	63.0	60.0	-
Kang *et al.* [21]	71.0	62.0	66.0	-
Zhang *et al.* [52]	83.0	67.0	74.0	0.48
Yao *et al.* [50]	76.5	75.3	75.9	1.61
EAST [53]	81.7	61.6	70.2	6.52
EAST † [53]	87.3	67.4	76.1	**13.2**
SegLink [39]	86.0	70.0	77.0	8.9
He *et al.* † [16]	77.0	70.0	74.0	1.1
ours	**87.6**	**76.2**	**81.5**	5.7

Table 4. Results on MSRA-TD500. † stands for the base net of the model is not VGG16.

finetuned another 240 epochs. In test stage, we input the images with the size 768 × 768 and set τ to 0.65.

As shown in Table 4, our method surpasses all the previous methods by a large margin. Our method achieves state-of-the-art performances both in recall, precision and F-measure (87.6%, 76.2% and 81.5%), and much better than the previous best result (81.5% *vs.* 77.0%). That means our method is more capable than other methods of detecting arbitrarily oriented long text.

5.6. Detecting Multi-Lingual Text

We verify the ability of our method to detect multi-lingual text on MLT. We finetune about 120 epochs on the model pre-trained on SynthText. When testing in single scale, the sizes of images are set as 768 × 768. We evaluate our method online and compare with some public results on the leaderboard [2]. As shown in Table 5, our method outperforms all competing methods by at least 3.1%.

5.7. Generalization Ability

To evaluate the generalization ability of our model, we test it on COCO-Text using the model finetuned on ICDAR2015. We set the test image size as 768 × 768. We use the annotations (V1.1) to compare with other methods, for the sake of fairness. The results are shown in Table 6. **Without training**, on COCO-Text, our method achieves an F-measure of 42.5%, better than competitors.

Method	Precision	Recall	F-measure
TH-DL [2]	67.8	34.8	46.0
SARI_FDU_RRPN_V1 [2]	71.2	55.5	62.4
Sensetime OCR [2]	56.9	69.4	62.6
SCUT_DLVClab1 [2]	80.3	54.5	65.0
e2e_ctc01_multi_scale [2]	79.8	61.2	69.3
ours	**83.8**	55.6	66.8
ours*	74.3	**70.6**	**72.4**

Table 5. Results on MLT. * means multi-scale.

Method	Precision	Recall	F-measure
Baseline A [46]	83.8	23.3	36.5
Baseline B [46]	**89.7**	10.7	19.1
Baseline C [46]	18.6	4.7	7.5
Yao *et al.* [50]	43.2	27.1	33.3
EAST [53]	50.4	**32.4**	39.5
WordSup [17]	45.2	30.9	36.8
SSTD [15]	46.0	31.0	37.0
ours	69.9	26.2	38.1
ours*	61.9	**32.4**	**42.5**
COCO-Text Challenge (IOU 0.5)			
UM [1]	47.6	**65.5**	55.1
TDN_SJTU_v2 [1]	62.4	54.3	58.1
Text_Detection_DL [1]	60.1	61.8	61.4
ours	**72.5**	52.9	61.1
ours*	62.9	62.2	**62.6**
COCO-Text Challenge (IOU 0.75)			
Text_Detection_DL [1]	25.2	25.5	25.4
UM [1]	22.7	31.2	26.3
TDN_SJTU_v2 [1]	31.8	27.7	29.6
ours	**40.0**	30.0	34.6
ours*	35.1	**34.8**	**34.9**

Table 6. Results on COCO-Text. * means multi-scale.

Besides, we also evaluate our model on the ICDAR2017 Robust Reading Challenge on COCO-Text [1] with the annotations V1.4. The results are reported in Table 6. Among all the public results in leaderboard [1], our method ranks the first. Especially when the threshold of iou is set to 0.75, the result that our method exceeds others in a large margin shows it can detect text more accurately.

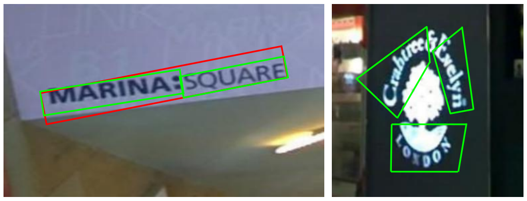


Figure 7. Failure cases of our method. The boxes in green are ground truth. The red boxes are our predictions.

5.8. Limitations

One limitation of the proposed method is that when two text instances are extremely close, it may predict the two instances as one (Fig. 7), since the position-sensitive segmentation might fail. Besides, the method is not good at detecting curved text (Fig. 7), as there are few curved samples in the training set.

6. Conclusion

In this paper, we have presented a scene text detector that localize text by corner point detection and position-sensitive segmentation. We evaluated it on several public benchmarks focusing on oriented, horizontal, long oriented and multi-lingual text. The superior performances demonstrate the effectiveness and robustness of our method. In the future, we are interested in constructing an end-to-end OCR system based on the proposed method.

References

[1] Coco-text challenge. `http://rrc.cvc.uab.es/?ch=5&com=evaluation&task=1`.

[2] Mlt-challenge. `http://rrc.cvc.uab.es/?ch=8&com=evaluation&task=1&gtv=1`.

[3] Pytorch. `http://pytorch.org/`.

[4] X. Bai, M. Yang, P. Lyu, Y. Xu, and J. Luo. Integrating scene text and visual appearance for fine-grained image classification. *arXiv preprint arXiv:1704.04613*, 2017.

[5] A. Bissacco, M. Cummins, Y. Netzer, and H. Neven. Photoocr: Reading text in uncontrolled conditions. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 785–792, 2013.

[6] M. Busta, L. Neumann, and J. Matas. Fastext: Efficient unconstrained scene text detector. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1206–1214, 2015.

[7] M. Busta, L. Neumann, and J. Matas. Deep textspotter: An end-to-end trainable scene text localization and recognition framework. In *Proc. ICCV*, 2017.

[8] J. Dai, K. He, Y. Li, S. Ren, and J. Sun. Instance-sensitive fully convolutional networks. In *European Conference on Computer Vision*, pages 534–549. Springer, 2016.

[9] J. Dai, Y. Li, K. He, and J. Sun. R-fcn: Object detection via region-based fully convolutional networks. In *Advances in neural information processing systems*, pages 379–387, 2016.

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

[11] C.-Y. Fu, W. Liu, A. Ranga, A. Tyagi, and A. C. Berg. Dssd: Deconvolutional single shot detector. *arXiv preprint arXiv:1701.06659*, 2017.

[12] R. Girshick. Fast r-cnn. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[13] L. Gómez and D. Karatzas. Textproposals: a text-specific selective search algorithm for word spotting in the wild. *Pattern Recognition*, 70:60–74, 2017.

[14] A. Gupta, A. Vedaldi, and A. 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.

[15] P. He, W. Huang, T. He, Q. Zhu, Y. Qiao, and X. Li. Single shot text detector with regional attention. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[16] W. He, X.-Y. Zhang, F. Yin, and C.-L. Liu. Deep direct regression for multi-oriented scene text detection. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[17] H. Hu, C. Zhang, Y. Luo, Y. Wang, J. Han, and E. Ding. Wordsup: Exploiting word annotations for character based text detection. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[18] L. Huang, Y. Yang, Y. Deng, and Y. Yu. Densebox: Unifying landmark localization with end to end object detection. *arXiv preprint arXiv:1509.04874*, 2015.

[19] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman. Reading text in the wild with convolutional neural networks. *International Journal of Computer Vision*, 116(1):1–20, 2016.

[20] M. Jaderberg, A. Vedaldi, and A. Zisserman. Deep features for text spotting. In *European conference on computer vision*, pages 512–528. Springer, 2014.

[21] L. Kang, Y. Li, and D. Doermann. Orientation robust text line detection in natural images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4034–4041, 2014.

[22] D. Karatzas, L. Gomez-Bigorda, A. Nicolaou, S. Ghosh, A. Bagdanov, M. Iwamura, J. Matas, L. Neumann, V. R. Chandrasekhar, S. Lu, et al. Icdar 2015 competition on robust reading. In *Document Analysis and Recognition (ICDAR), 2015 13th International Conference on*, pages 1156–1160. IEEE, 2015.

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

[24] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[25] H. Li, P. Wang, and C. Shen. Towards end-to-end text spotting with convolutional recurrent neural networks. In *The*

IEEE International Conference on Computer Vision (ICCV), Oct 2017.

[26] Y. Li, H. Qi, J. Dai, X. Ji, and Y. Wei. Fully convolutional instance-aware semantic segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[27] M. Liao, B. Shi, X. Bai, X. Wang, and W. Liu. Textboxes: A fast text detector with a single deep neural network. In *AAAI*, pages 4161–4167, 2017.

[28] T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[29] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[30] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[31] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[32] J. Ma, W. Shao, H. Ye, L. Wang, H. Wang, Y. Zheng, and X. Xue. Arbitrary-oriented scene text detection via rotation proposals. *arXiv preprint arXiv:1703.01086*, 2017.

[33] F. Milletari, N. Navab, and S.-A. Ahmadi. V-net: Fully convolutional neural networks for volumetric medical image segmentation. In *3D Vision (3DV), 2016 Fourth International Conference on*, pages 565–571. IEEE, 2016.

[34] L. Neumann and J. Matas. A method for text localization and recognition in real-world images. In *Asian Conference on Computer Vision*, pages 770–783. Springer, 2010.

[35] L. Neumann and J. Matas. Efficient scene text localization and recognition with local character refinement. In *Document Analysis and Recognition (ICDAR), 2015 13th International Conference on*, pages 746–750. IEEE, 2015.

[36] L. Neumann and J. Matas. Real-time lexicon-free scene text localization and recognition. *IEEE transactions on pattern analysis and machine intelligence*, 38(9):1872–1885, 2016.

[37] J. Redmon, S. Divvala, R. Girshick, and A. 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.

[38] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[39] B. Shi, X. Bai, and S. Belongie. Detecting oriented text in natural images by linking segments. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[40] A. Shrivastava, A. Gupta, and R. Girshick. Training region-based object detectors with online hard example mining. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 761–769, 2016.

[41] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[42] S. Tian, S. Lu, and C. Li. Wetext: Scene text detection under weak supervision. *arXiv preprint arXiv:1710.04826*, 2017.

[43] S. Tian, Y. Pan, C. Huang, S. Lu, K. Yu, and C. 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.

[44] Z. Tian, W. Huang, T. He, P. He, and Y. Qiao. Detecting text in natural image with connectionist text proposal network. In *European Conference on Computer Vision*, pages 56–72. Springer, 2016.

[45] L. Tychsen-Smith and L. Petersson. Denet: Scalable real-time object detection with directed sparse sampling. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[46] A. Veit, T. Matera, L. Neumann, J. Matas, and S. Belongie. Coco-text: Dataset and benchmark for text detection and recognition in natural images. *arXiv preprint arXiv:1601.07140*, 2016.

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

[48] X. Wang, K. Chen, Z. Huang, C. Yao, and W. Liu. Point linking network for object detection. *arXiv preprint arXiv:1706.03646*, 2017.

[49] C. Yao, X. Bai, W. Liu, Y. Ma, and Z. Tu. Detecting texts of arbitrary orientations in natural images. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 1083–1090. IEEE, 2012.

[50] C. Yao, X. Bai, N. Sang, X. Zhou, S. Zhou, and Z. Cao. Scene text detection via holistic, multi-channel prediction. *arXiv preprint arXiv:1606.09002*, 2016.

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

[52] Z. Zhang, C. Zhang, W. Shen, C. Yao, W. Liu, and X. 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.

[53] X. Zhou, C. Yao, H. Wen, Y. Wang, S. Zhou, W. He, and J. Liang. East: An efficient and accurate scene text detector. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.