Detecting Oriented Text in Natural Images by Linking Segments

Baoguang Shi[1]   Xiang Bai[1]*   Serge Belongie[2]

[1]School of EIC, Huazhong University of Science and Technology
[2]Department of Computer Science, Cornell Tech

shibaoguang@gmail.com   xbai@hust.edu.cn   sjb344@cornell.edu

Abstract

Most state-of-the-art text detection methods are specific to horizontal Latin text and are not fast enough for real-time applications. We introduce Segment Linking (SegLink), an oriented text detection method. The main idea is to decompose text into two locally detectable elements, namely segments and links. A segment is an oriented box covering a part of a word or text line; A link connects two adjacent segments, indicating that they belong to the same word or text line. Both elements are detected densely at multiple scales by an end-to-end trained, fully-convolutional neural network. Final detections are produced by combining segments connected by links. Compared with previous methods, SegLink improves along the dimensions of accuracy, speed, and ease of training. It achieves an f-measure of 75.0% on the standard ICDAR 2015 Incidental (Challenge 4) benchmark, outperforming the previous best by a large margin. It runs at over 20 FPS on 512×512 images. Moreover, without modification, SegLink is able to detect long lines of non-Latin text, such as Chinese.

1. Introduction

Reading text in natural images is a challenging task under active research. It is driven by many real-world applications, such as Photo OCR [2], geo-location, and image retrieval [9]. In a text reading system, text detection, *i.e.* localizing text with bounding boxes of words or text lines, is usually the first step of great significance. In a sense, text detection can be seen as object detection applied to text, where words/characters/text lines are taken as the detection targets. Owing to this, a new trend has emerged recently that state-of-the-art text detection methods [9, 6, 22, 30] are heavily based on the advanced general object detection or segmentation techniques, *e.g.* [4, 5, 15].

Despite the great success of the previous work, we argue that the general detection methods are not well suited
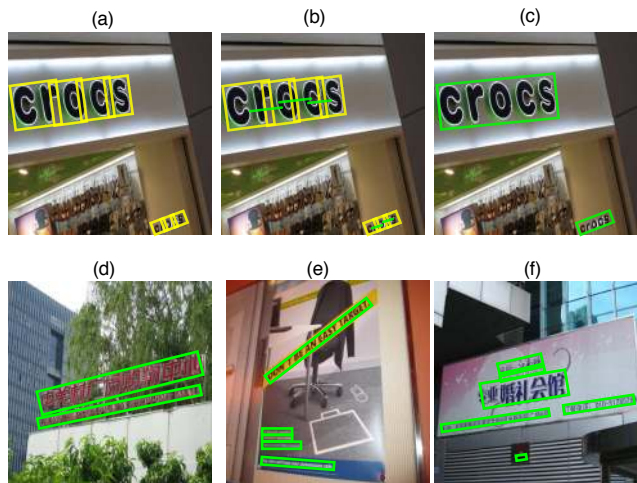

Figure 1. **SegLink Overview.** The upper row shows an image with two words of different scales and orientations. (a) Segments (yellow boxes) are detected on the image. (b) Links (green lines) are detected between pairs of adjacent segments. (c) Segments connected by links are combined into whole words. (d-f) SegLink is able to detect long lines of Latin and non-Latin text, such as Chinese.

for text detection, for two main reasons. First, word/text line bounding boxes have much larger aspect ratios than those of general objects. An (fast/faster) R-CNN [5, 4, 19]- or SSD [14]-style detector may suffer from the difficulty of producing such boxes, owing to its proposal or anchor box design. In addition, some non-Latin text does not have blank spaces between words, hence the even larger bounding box aspect ratios, which make the problem worse. Second, unlike general objects, text usually has a clear definition of orientation [25]. It is important for a text detector to produce oriented boxes. However, most general object detection methods are not designed to produce oriented boxes.

To overcome the above challenges, we tackle the text detection problem in a new perspective. We propose to decompose long text into two smaller and locally-detectable elements, namely *segment* and *link*. As illustrated in Fig. 1, a segment is an oriented box that covers a part of a word
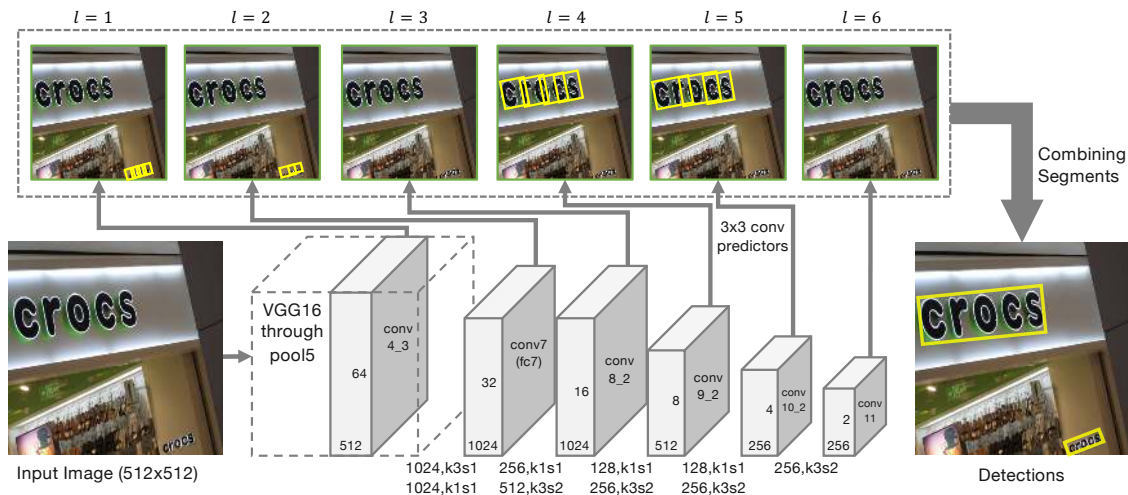
---

*Corresponding author.

Figure 2. Network Architecture. The network consists of convolutional feature layers (shown as gray blocks) and convolutional predictors (thin gray arrows). Convolutional filters are specified in the format of "(#filters),k(kernel size)s(stride)". A multi-line filter specification means a hidden layer between. Segments (yellow boxes) and links (not displayed) are detected by convolutional predictors on multiple feature layers (indexed by $l = 1 \ldots 6$) and combined into whole words by a combining algorithm.

(for clarity we use "word" here and later on, but segments also work seamlessly on text lines that comprise multiple words); A link connects a pair of adjacent segments, indicating that they belong to the same word. Under the above definitions, a word is located by a number of segments with links between them. During detection, segments and links are densely detected on an input image by a convolutional neural network. Then, the segments are combined into whole words according to the links.

The key advantage of this approach is that long and oriented text is now detected locally since both basic elements are *locally-detectable*: Detecting a segment does not require the whole word to be observed. And neither does a link since the connection of two segments can be inferred from a local context. Thereafter, we can detect text of any length and orientation with great flexibility and efficiency.

Concretely, we propose a convolutional neural network (CNN) model to detect both segments and links simultaneously, in a *fully-convolutional* manner. The network uses VGG-16 [21] as its backbone. A few extra feature layers are added onto it. Convolutional predictors are added to 6 of the feature layers to detect segments and links at different scales. To deal with redundant detections, we introduce two types of links, namely *within-layer links* and *cross-layer links*. A within-layer link connects a segment to its neighbors on the same layer. A cross-layer link, on the other hand, connects a segment to its neighbors on the lower layer. In this way, we connect segments of adjacent locations as well as scales. Finally, we find connected segments with a depth-first search (DFS) algorithm and combine them into whole words.

Our main contribution is the novel segment-linking de-

tection method. Through experiments, we show that the proposed method possesses several distinctive advantages over the other state-of-the-art methods: 1) *Robustness*: SegLink models the structure of oriented text in a simple and elegant way, with robustness against complex backgrounds. Our method achieves highly competitive results on standard datasets. In particular, it outperforms the previous best by a large margin in terms of f-measure (75.0% vs 64.8%) on the ICDAR 2015 Incidental (Challenge 4) benchmark [12]; 2) *Efficiency*: SegLink is highly efficient due to its single-pass, fully-convolutional design. It processes more than 20 images of 512x512 size per second; 3) *Generality*: Without modification, SegLink is able to detect long lines of non-Latin text, such as Chinese. We demonstrate this capability on a multi-lingual dataset.

2. Related Work

Text Detection Over the past few years, much research effort has been devoted to the text detection problem [24, 23, 17, 17, 25, 7, 8, 30, 29, 2, 9, 6, 22, 26]. Based on the basic detection targets, the previous methods can be roughly divided into three categories: *character-based*, *word-based* and *line-based*. Character-based methods [17, 23, 24, 10, 7, 8] detect individual characters and group them into words. These methods find characters by classifying candidate regions extracted by region extraction algorithms or by classifying sliding windows. Such methods often involve a post-processing step of grouping characters into words. Word-based methods [9, 6] directly detect word bounding boxes. They often have a similar pipeline to the recent CNN-based general object detection networks. Though achieving excellent detection accuracies, these methods may suffer from

performance drop when applied to some non-Latin text such as Chinese, as we mentioned earlier. Line-based methods [29, 30, 26] find text regions using some image segmentation algorithms. They also require a sophisticated post-processing step of word partitioning and/or false positive removal. Compared with the previous approaches, our method predicts segments and links jointly in a single forward network pass. The pipeline is much simpler and cleaner. Moreover, the network is end-to-end trainable.

Our method is similar in spirit to a recent work [22], which detects text lines by finding and grouping a sequence of *fine-scale text proposals* through a CNN coupled with recurrent neural layers. In contrast, we detect oriented segments only using convolutional layers, yielding better flexibility and faster speed. Also, we detect links explicitly using the same strong CNN features for segments, improving the robustness.

Object Detection   Text detection can be seen as a particular instance of general object detection, which is a fundamental problem in computer vision. Most state-of-the-art detection systems either classify some class-agnostic object proposals with CNN [5, 4, 19] or directly regress object bounding boxes from a set of preset boxes (*e.g.* anchor boxes) [18, 14].

The architecture of our network inherits that of SSD [14], a recent object detection model. SSD proposed the idea of detecting objects on multiple feature layers with convolutional predictors. Our model also detects segments and links in a very similar way. Despite the model similarity, our detection strategy is drastically different: SSD directly outputs object bounding boxes. We, on the other hand, adopt a bottom-up approach by detecting the two comprising elements of a word or text line and combine them together.

3. Segment Linking

Our method detects text with a feed-forward CNN model. Given an input image I of size $w_I \times h_I$, the model outputs a fixed number of segments and links, which are then filtered by their confidence scores and combined into whole word bounding boxes. A bounding box is a rotated rectangle denoted by $b = (x_b, y_b, w_b, h_b, \theta_b)$, where x_b, y_b are the coordinates of the center, w_b, h_b the width and height, and θ_b the rotation angle.

3.1. CNN Model

Fig. 2 shows the network architecture. Our network uses a pretrained VGG-16 network [21] as its backbone (conv1 through pool5). Following [14], the fully-connected layers of VGG-16 are converted into convolutional layers (fc6 to conv6; fc7 to conv7). They are followed by a few extra convolutional layers (conv8_1 to conv11), which extract even deeper features with larger receptive fields. Their configurations are specified in Fig. 2.

Segments and links are detected on 6 of the feature layers, which are conv4_3, conv7, conv8_2, conv9_2, conv10_2, and conv11. These feature layers provide high-quality deep features of different granularity (conv4_3 the finest and conv11 the coarsest). A convolutional predictor with 3×3 kernels is added to each of the 6 layers to detect segments and links. We index the feature layers and the predictors by $l = 1, \ldots, 6$.

Segment Detection   Segments are also oriented boxes, denoted by $s = (x_s, y_s, w_s, h_s, \theta_s)$. We detect segments by estimating the confidence scores and geometric offsets to a set of *default boxes* [14] on the input image. Each default box is associated with a feature map location, and its score and offsets are predicted from the features at that location. For simplicity, we only associate one default box with a feature map location.

Consider the l-th feature layer whose feature map size is $w_l \times h_l$. A location (x, y) on this map corresponds to a default box centered at (x_a, y_a) on the image, where

$$x_a = \frac{w_I}{w_l}(x + 0.5); \quad y_a = \frac{h_I}{h_l}(y + 0.5) \qquad (1)$$

The width and height of the default box are both set to a constant a_l.

The convolutional predictor produces 7 channels for segment detection. Among them, 2 channels are further softmax-normalized to get the segment score in $(0, 1)$. The rest 5 are the geometric offsets. Considering a location (x, y) on the map, we denote the vector at this location along the depth by $(\Delta x_s, \Delta y_s, \Delta w_s, \Delta h_s, \Delta \theta_s)$. Then, the segment at this location is calculated by:

$$x_s = a_l \Delta x_s + x_a \qquad (2)$$
$$y_s = a_l \Delta y_s + y_a \qquad (3)$$
$$w_s = a_l \exp(\Delta w_s) \qquad (4)$$
$$h_s = a_l \exp(\Delta h_s) \qquad (5)$$
$$\theta_s = \Delta \theta_s \qquad (6)$$

Here, the constant a_l controls the scale of the output segments. It should be chosen with regard to the receptive field size of the l-th layer. We use an empirical equation for choosing this size: $a_l = \gamma \frac{w_I}{w_l}$, where $\gamma = 1.5$.

Within-Layer Link Detection   A link connects a pair of adjacent segments, indicating that they belong to the same word. Here, adjacent segments are those detected at adjacent feature map locations. Links are not only necessary for combining segments into whole words but also helpful for separating two nearby words – between two nearby words, the links should be predicted as negative.
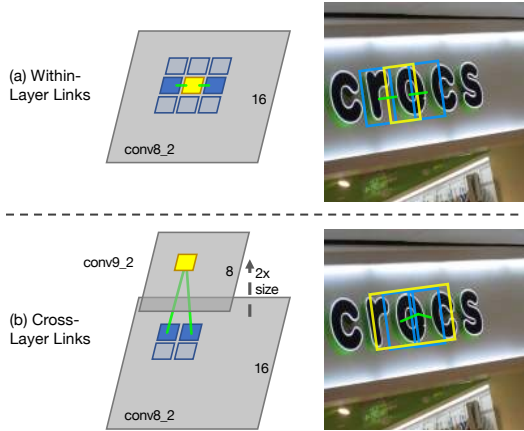
Figure 3. **Within-Layer and Cross-Layer Links.** (a) A location on conv8_2 (yellow block) and its 8-connected neighbors (blue blocks with and without fill). The detected within-layer links (green lines) connect a segment (yellow box) and its two neighboring segments (blue boxes) on the same layer. (b) The cross-layer links connect a segment on conv9_2 (yellow box) and two segments on conv8_2 (blue boxes).

We explicitly detect links between segments using the same features for detecting segments. Since we detect only one segment at a feature map location, segments can be indexed by their map locations (x, y) and layer indexes l, denoted by $s^{(x,y,l)}$. As illustrated in Fig. 3.a, we define the *within-layer neighbors* of a segment as its 8-connected neighbors on the same feature layer:

$$\mathcal{N}^w_{s^{(x,y,l)}} = \left\{ s^{(x',y',l)} \right\}_{x-1 \le x' \le x+1, y-1 \le y' \le y+1} \setminus s^{(x,y,l)}$$

$$(7)$$

As segments are detected locally, a pair of neighboring segments are also adjacent on input image. Links are also detected by the convolutional predictors. A predictor outputs 16 channels for the links to the 8-connected neighboring segments. Every 2 channels are softmax-normalized to get the score of a link.

Cross-Layer Link Detection   In our network, segments are detected at different scales on different feature layers. Each layer handles a range of scales. We make these ranges overlap in order not to miss scales at their edges. But as a result, segments of the same word could be detected on multiple layers at the same time, producing redundancies.

To address this problem, we further propose another type of links, called *cross-layer links*. A cross-layer link connects segments on two feature layers with adjacent indexes. For example, cross-layer links are detected between conv4_3 and conv7, because their indexes are $l = 1$ and $l = 2$ respectively.

An important property of such a pair is that the first layer always has twice the size as the second one, because of the

down-sampling layer (max-pooling or stride-2 convolution) between them. Note that this property only holds when all feature layers have even-numbered sizes. In practice, we ensured this property by having the width and height of the input image both dividable by 128. For example, an 1000×800 image is resized to 1024×768, which is the nearest valid size.

As illustrated in Fig. 3.b, we define the *cross-layer neighbors* of a segment as

$$\mathcal{N}^c_{s^{(x,y,l)}} = \left\{ s^{(x',y',l-1)} \right\}_{2x \le x' \le 2x+1, 2y \le y' \le 2y+1}, \quad (8)$$

which are the segments on the preceeding layer. Every segment has 4 cross-layer neighbors. The correspondence is ensured by the double-size relationship between the two layers.

Again, cross-layer links are detected by the convolutional predictor. The predictor outputs 8 channels for cross-layer links. Every 2 channels are softmax-normalized to produce the score of a cross-layer link. Cross-layer links are detected on feature layer $l = 2 \ldots 6$, but not on $l = 1$ (conv4_3) since it has no preceeding feature layer.

With cross-layer links, segments of different scales can be connected and later combined. Compared with the traditional non-maximum suppression, cross-layer linking provides a trainable way of joining redundancies. Besides, it fits seamlessly into our linking strategy and is easy to implement under our framework.
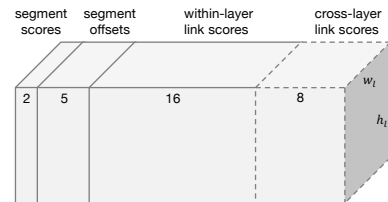

Figure 4. Output channels of a convolutional predictor. The block shows a $w_l \times h_l$ map of depth 31. The predictor of $l = 1$ does not output the channels for corss-layer links.

Outputs of a Convolutional Predictor   Putting things together, Fig. 4 shows the output channels of a convolutional predictor. A predictor is implemented by a convolutional layer followed by some softmax layers that normalize the segment and link scores respectively. Thereafter, all layers in our network are convolutional layers. Our network is fully-convolutional.

3.2. Combining Segments with Links

After feed-forwarding, the network produces a number of segments and links (the number depends on the image size). Before combination, the output segments and links are filtered by their confidence scores. We set different filtering thresholds for segment and link, respectively α and β.

Empirically, the performance of our model is not very sensitive to these thresholds. A 0.1 deviation on either thresholds from their optimal values results in less than 1% f-measure drop.

Taking the filtered segments as nodes and the filtered links as edges, we construct a graph over them. Then, a depth-first search (DFS) is performed over the graph to find its connected components. Each component contains a set of segments that are connected by links. Denoting a connected component by $\mathcal{B}$, segments within this component are combined following the procedures in Alg. 1.

Algorithm 1 Combining Segments

1: **Input:** $\mathcal{B} = \{s^{(i)}\}_{i=1}^{|\mathcal{B}|}$ is a set of segments connected by links, where $s^{(i)} = (x_s^{(i)}, y_s^{(i)}, w_s^{(i)}, h_s^{(i)}, \theta_s^{(i)})$.
2: Find the average angle $\theta_b := \frac{1}{|\mathcal{B}|} \sum_{\mathcal{B}} \theta_s^{(i)}$.
3: For a straight line $(\tan \theta_b) x + b$, find the b that minimizes the sum of distances to all segment centers $(x_s^{(i)}, y_s^{(i)})$.
4: Find the perpendicular projections of all segment centers onto the straight line.
5: From the projected points, find the two with the longest distance. Denote them by (x_p, y_p) and (x_q, y_q).
6: $x_b := \frac{1}{2}(x_p + x_q)$
7: $y_b := \frac{1}{2}(y_p + y_q)$
8: $w_b := \sqrt{(x_p - x_q)^2 + (y_p - y_q)^2} + \frac{1}{2}(w_p + w_q)$
9: $h_b := \frac{1}{|\mathcal{B}|} \sum_{\mathcal{B}} h_s^{(i)}$
10: $b := (x_b, y_b, w_b, h_b, \theta_b)$
11: **Output:** b is the combined bounding box.

---

4. Training

4.1. Groundtruths of Segments and Links

The network is trained by the direct supervision of groundtruth segments and links. The groundtruths include the labels of all default boxes (*i.e.* the label of their corresponding segments), their offsets to the default boxes, and the labels of all within- and cross-layer links. We calculate them from the groundtruth word bounding boxes.

First, we assume that there is only one groundtruth word on the input image. A default box is labeled as positive iff 1) the center of the box is inside the word bounding box; 2) the ratio between the box size a_l and the word height h satisfies:

$$\max(\frac{a_l}{h}, \frac{h}{a_l}) \leq 1.5 \qquad (9)$$

Otherwise, the default box is labeled as negative.

Next, we consider the case of multiple words. A default box is labeled as negative if it does not meet the above-mentioned criteria for any word. Otherwise, it is labeled as

positive and matched to the word that has the closest size, *i.e.* the one with the minimal value at the left-hand side of Eq. 9.

(1) Default box, word bounding box, and the center of the default box (blue dot)

(2) Rotate word clockwise by θ along the center of the default box

(3) Crop word bounding box to remove the parts to the left and right of the default box

(4) Rotate the cropped box anticlockwise by θ along the center of the default box
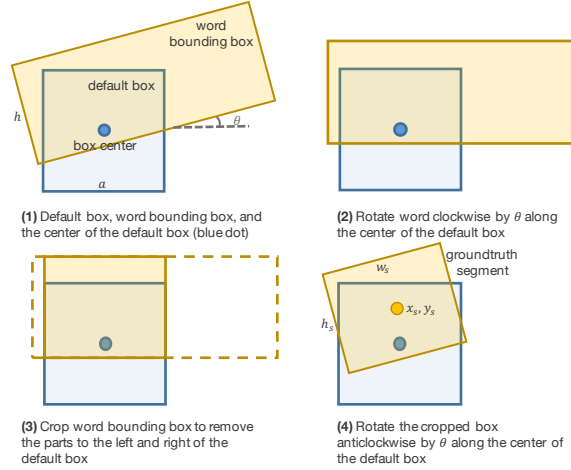
Figure 5. The steps of calculating a groundtruth segment given a default box and a word bounding box.

Offsets are calculated on positive default boxes. First, we calculate the groundtruth segments following the steps illustrated in Fig. 5. Then, we solve Eq. 2 to Eq. 6 to get the groundtruth offsets.

A link (either within-layer or cross-layer) is labeled as positive iff 1) both of the default boxes connected to it are labeled as positive; 2) the two default boxes are matched to the same word.

4.2. Optimization

Objective Our network model is trained by simultaneously minimizing the losses on segment classification, offsets regression, and link classification. Overall, the loss function is a weighted sum of the three losses:

$$L(\mathbf{y}_s, \mathbf{c}_s, \mathbf{y}_l, \mathbf{c}_l, \hat{\mathbf{s}}, \mathbf{s}) = \frac{1}{N_s} L_{\text{conf}}(\mathbf{y}_s, \mathbf{c}_s) + \lambda_1 \frac{1}{N_s} L_{\text{loc}}(\hat{\mathbf{s}}, \mathbf{s})$$
$$+ \lambda_2 \frac{1}{N_l} L_{\text{conf}}(\mathbf{y}_l, \mathbf{c}_l) \qquad (10)$$

Here, $\mathbf{y}_s$ is the labels of all segments. $\mathbf{y}_s^{(i)} = 1$ if the i-th default box is labeled as positive, and 0 otherwise. Likewise, $\mathbf{y}_l$ is the labels of the links. L_{conf} is the softmax loss over the predicted segment and link scores, respectively $\mathbf{c}_s$ and $\mathbf{c}_l$. L_{loc} is the *Smooth L1* regression loss [4] over the predicted segment geometries $\hat{\mathbf{s}}$ and the groundtruth $\mathbf{s}$. The losses on segment classification and regression are normalized by N_s, which is the number of positive default boxes. The loss on link classification is normalized by the number of positive links N_l. The weight constants λ_1 and λ_2 are both set to 1 in practice.

Online Hard Negative Mining For both segments and links, negatives take up most of the training samples. Therefore, hard negative mining is necessary for balancing the positive and negative samples. We follow the online hard negative mining strategy proposed in [20] to keep the ratio between the negatives and positives 3:1 at most. Hard negative mining is performed separately for segments and links.

Data Augmentation We adopt an online augmentation pipeline that is similar to that of SSD [14] and YOLO [18]. Training images are randomly cropped to a patch that has a minimum Jaccard overlap of o with any groundtruth word Crops are resized to the same size before loaded into a batch. For oriented text, the augmentation is performed on the axis-aligned bounding boxes of the words. The overlap o is randomly chosen from 0 (no constraint), 0.1, 0.3, 0.5, 0.7, and 0.9 for every sample. The crop size is randomly chosen from $[0.1, 1]$ of the original image size. Training images are *not* horizontally flipped.

5. Experiments

In this section, we evaluate the proposed method on three public datasets, namely ICDAR 2015 Incidental Text (Challenge 4), MSRA-TD500, and ICDAR 2013, using the standard evaluation protocol of each.

5.1. Datasets

SynthText in the Wild (SynthText) [6] contains 800,000 *synthetic* training images. They are created by blending natural images with text rendered with random fonts, size, orientation, and color. Text is rendered and aligned to carefully chosen image regions in order have a realistic look. The dataset provides very detailed annotations for characters, words, and text lines. We only use the dataset for pre-training our network.

ICDAR 2015 Incidental Text (IC15) [12] is the Challenge 4 of the ICDAR 2015 Robust Reading Competition. This challenge features *incidental* scene text images taken by Google Glasses without taking care of positioning, image quality, and viewpoint. Consequently, the dataset exhibits large variations in text orientation, scale, and resolution, making it much more difficult than previous ICDAR challenges. The dataset contains 1000 training images and 500 testing images. Annotations are provided as word quadrilaterals.

MSRA-TD500 (TD500) [25] is the first standard dataset that focuses on oriented text. The dataset is also multilingual, including both Chinese and English text. The

dataset consists of 300 training images and 200 testing images. Different from IC15, TD500 is annotated at the level of text lines.

ICDAR 2013 (IC13) [13] contains mostly horizontal text, with some text slightly oriented. The dataset has been widely adopted for evaluating text detection methods previously. It consists of 229 training images and 233 testing images.

5.2. Implementation Details

Our network is pre-trained on SynthText and finetuned on real datasets (specified later). It is optimized by the standard SGD algorithm with a momentum of 0.9. For both pre-training and finetuning, images are resized to 384×384 after random cropping. Since our model is fully-convolutional, we can train it on a certain size and apply it to other sizes during testing. Batch size is set to 32. In pretraining, the learning is set to 10^{-3} for the first 60k iterations, then decayed to 10^{-4} for the rest 30k iterations. During finetuning, the learning rate is fixed to 10^{-4} for 5-10k iterations. The number of finetuning iterations depends on the size of the dataset.

Due to the precision-recall tradeoff and the difference between evaluation protocols across datasets, we choose the best thresholds α and β to optimize f-measure. Except for IC15, the thresholds are chosen separately on different datasets via a grid search with 0.1 step on a hold-out validation set. IC15 does not offer an offline evaluation script, so the only way for us is to submit multiple results to the evaluation server.

Our method is implemented using TensorFlow [1] r0.11. All the experiments are carried out on a workstation with an Intel Xeon 8-core CPU (2.8 GHz), 4 Titan X Graphics Cards, and 64GB RAM. Running on 4 GPUs in parallel, training a batch takes about 0.5s. The whole training process takes less than a day.

5.3. Detecting Oriented English Text

First, we evaluate SegLink on IC15. The pretrained model is finetuned for 10k iterations on the training dataset of IC15. Testing images are resized to 1280×768. We set the thresholds on segments and links to 0.9 and 0.7, respectively. Performance is evaluated by the official central submission server (http://rrc.cvc.uab.es/?ch=4). In order to meet the requirements on submission format, the output oriented rectangles are converted into quadrilaterals.

Table 1 lists and compares the results of the proposed method and other state-of-the-art methods. Some results are obtained from the online leaderboard. SegLink outperforms the others by a large margin. In terms of f-measure, it outperforms the second best by 10.2%. Considering that some methods have close or even higher precision than SegLink,

Recall=1.0 Precision=0.86 F-Score=0.92   Recall=1.0 Precision=1.0 F-Score=1.0   Recall=1.0 Precision=0.88 F-Score=0.93

Recall=1.0 Precision=1.0 F-Score=1.0   Recall=0.88 Precision=0.88 F-Score=0.88   Recall=1.0 Precision=1.0 F-Score=1.0

Figure 6. **Example Results on IC15.** Green regions are correctly detected text regions. Red ones are either false positive or false negative. Gray ones are detected but neglected by the evaluation algorithm. Visualizations are generated by the central submission system. Yellow frames contain zoom-in image regions.

Table 1. Results on ICDAR 2015 Incidental Text

Method	Precision	Recall	F-measure
HUST_MCLAB	47.5	34.8	40.2
NJU_Text	72.7	35.8	48.0
StradVision-2	**77.5**	36.7	49.8
MCLAB_FCN [30]	70.8	43.0	53.6
CTPN [22]	51.6	74.2	60.9
Megvii-Image++	72.4	57.0	63.8
Yao *et al.* [26]	72.3	58.7	64.8
SegLink	73.1	**76.8**	**75.0**

Table 2. Results on MSRA-TD500

Method	Precision	Recall	F-measure	FPS
Kang *et al.* [11]	71	62	66	-
Yao *et al.* [25]	63	63	60	0.14
Yin *et al.* [27]	81	63	74	0.71
Yin *et al.* [28]	71	61	65	1.25
Zhang *et al.* [30]	83	67	74	0.48
Yao *et al.* [26]	77	**75**	76	∼1.61
SegLink	**86**	70	**77**	**8.9**

the improvement mainly comes from the recall. As shown in Fig. 6, our method is able to distinguish text from very cluttered backgrounds. In addition, owing to its explicit link prediction, SegLink correctly separates words that are very close to each other.

5.4. Detecting Multi-Lingual Text in Long Lines

We further demonstrate the ability of SegLink to detect long text in non-Latin scripts. TD500 is taken as the dataset for this experiment, as it consists of oriented and multi-lingual text. The training set of TD500 only has 300 images, which are not enough for finetuning our model. We mix the training set of TD500 with the training set of IC15, in the way that every batch has half of its images coming from each dataset. The pretrained model is finetuned for 8k iterations. The testing images are resized to 768×768. The thresholds α and β are set to 0.9 and 0.5 respectively. Performance scores are calculated by the official development toolkit.

According to Table 2, SegLink achieves the highest scores in terms of precision and f-measure. Benefiting from its fully-convolutional design, SegLink runs at 8.9 FPS, a

much faster speed than the others. SegLink also enjoys simplicity. The inference process of SegLink is a single forward pass in the detection network, while the previous methods [25, 28, 30] involve sophisticated rule-based grouping or filtering steps.

TD500 contains many long lines of text in mixed languages (English and Chinese). Fig. 7 shows how SegLink handles such text. As can be seen, segments and links are densely detected along text lines. They result in long bounding boxes that are hard to obtain from a conventional object detector. Despite the large difference in appearance between English and Chinese text, SegLink is able to handle them simultaneously without any modifications in its structure.

5.5. Detecting Horizontal Text

Lastly, we evaluate the performance of SegLink on horizontal-text datasets. The pretrained model is finetuned for 5k iterations on the combined training sets of IC13 and IC15. Since the most text in IC13 has relatively larger sizes, the testing images are resized to 512×512. The thresholds α and β are set to 0.6 and 0.3, respectively. To match the submission format, we convert the detected oriented boxes
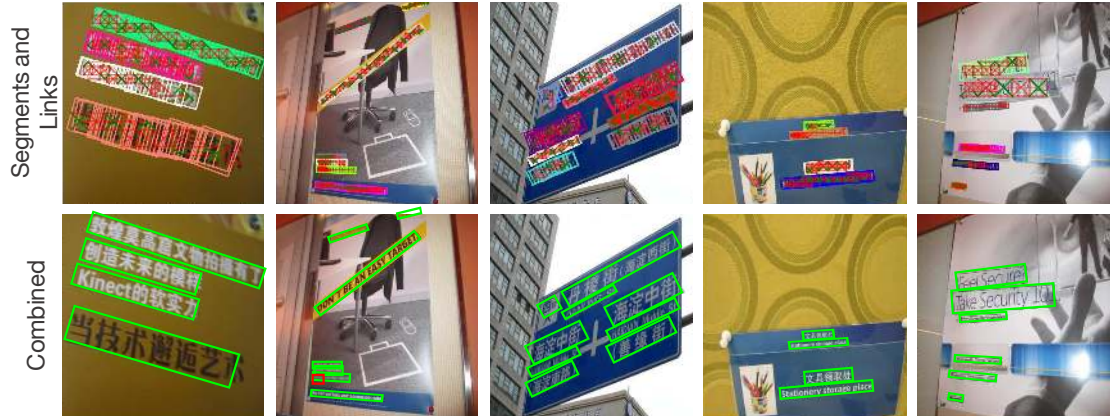
Figure 7. **Example Results on TD500.** The first row shows the detected segments and links. The within-layer and cross-layer links are visualized as red and green lines, respectively. Segments are shown as rectangles in different colors, denoting different connected components. The second row shows the combined boxes.

into their axis-aligned bounding boxes.

Table 3 compares SegLink with other state-of-the-art methods. The scores are calculated by the central submission system using the "Deteval" evaluation protocol. SegLink achieves very competitive results in terms of f-measure. Only one approach [22] outperforms SegLink in terms of f-measure. However, [22] is mainly designed for detecting horizontal text and is not well-suited for oriented text. In terms of speed, SegLink runs at over 20 FPS on 512×512 images, much faster than the other methods.

Table 3. **Results on IC13**. P, R, F stand for precision, recall and f-measure respectively. *These methods are only evaluated under the "ICDAR 2013" evaluation protocol, the rest under "Deteval". The two protocols usually yield very close scores.

Method	P	R	F	FPS
Neumann *et al.* [16]*	81.8	72.4	77.1	3
Neumann *et al.* [17]*	82.1	71.3	76.3	3
Busta *et al.* [3]*	84.0	69.3	76.8	6
Zhang *et al.* [29]	88	74	80	<0.1
Zhang *et al.* [30]	88	78	83	<1
Jaderberg *et al.* [9]	88.5	67.8	76.8	<1
Gupta *et al.* [6]	92.0	75.5	83.0	15
Tian *et al.* [22]	**93.0**	**83.0**	**87.7**	7.1
SegLink	87.7	**83.0**	85.3	**20.6**

5.6. Limitations

A major limitation of SegLink is that two thresholds, α and β, need to be set manually. In practice, the optimal values of the thresholds are found by a grid search. Simplifying the parameters would be part of our future work. Another weakness is that SegLink fails to detect text that has very large character spacing. Fig. 8.a,b show two such cases. The detected links connect adjacent segments but fail to link distant segments.

Figure 8. **Failure cases on TD500**. Red boxes are false positives. (a)(b) SegLink fails to link the characters with large character spacing. (c) SegLink fails to detect curved text.

Fig. 8.c shows that SegLink fails to detect text of curved shape. However, we believe that this is not a limitation of the segment linking strategy, but the segment combination algorithm, which can only produce rectangles currently.

6. Conclusion

We have presented SegLink, a novel text detection strategy implemented by a simple and highly-efficient CNN model. The superior performance on horizontal, oriented, and multi-lingual text datasets well demonstrate that SegLink is accurate, fast, and flexible. In the future, we will further explore its potentials on detecting deformed text such as curved text. Also, we are interested in extending SegLink into a end-to-end recognition system.

Acknowledgment

References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org. 6

[2] A. Bissacco, M. Cummins, Y. Netzer, and H. Neven. Photoocr: Reading text in uncontrolled conditions. In *ICCV*, 2013. 1, 2

[3] M. Busta, L. Neumann, and J. Matas. Fastext: Efficient unconstrained scene text detector. In *ICCV*, 2015. 8

[4] R. B. Girshick. Fast R-CNN. In *ICCV*, 2015. 1, 3, 5

[5] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014. 1, 3

[6] A. Gupta, A. Vedaldi, and A. Zisserman. Synthetic data for text localisation in natural images. In *CVPR*, 2016. 1, 2, 6, 8

[7] W. Huang, Z. Lin, J. Yang, and J. Wang. Text localization in natural images using stroke feature transform and text covariance descriptors. In *ICCV*, 2013. 2

[8] W. Huang, Y. Qiao, and X. Tang. Robust scene text detection with convolution neural network induced MSER trees. In *ECCV*, 2014. 2

[9] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman. Reading text in the wild with convolutional neural networks. *IJCV*, 116(1):1–20, 2016. 1, 2, 8

[10] M. Jaderberg, A. Vedaldi, and A. Zisserman. Deep features for text spotting. In *ECCV*, 2014. 2

[11] L. Kang, Y. Li, and D. S. Doermann. Orientation robust text line detection in natural images. In *CVPR*, 2014. 7

[12] D. Karatzas, L. Gomez-Bigorda, A. Nicolaou, S. K. Ghosh, A. D. Bagdanov, M. Iwamura, J. Matas, L. Neumann, V. R. Chandrasekhar, S. Lu, F. Shafait, S. Uchida, and E. Valveny. ICDAR 2015 competition on robust reading. In *ICDAR 2015*, 2015. 2, 6

[13] D. Karatzas, F. Shafait, S. Uchida, M. Iwamura, L. G. i Bigorda, S. R. Mestre, J. Mas, D. F. Mota, J. Almazán, and L. de las Heras. ICDAR 2013 robust reading competition. In *ICDAR 2013*, 2013. 6

[14] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C. Fu, and A. C. Berg. SSD: single shot multibox detector. In *ECCV*, pages 21–37, 2016. 1, 3, 6

[15] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015. 1

[16] L. Neumann and J. Matas. Efficient scene text localization and recognition with local character refinement. In *ICDAR*, 2015. 8

[17] L. Neumann and J. Matas. Real-time lexicon-free scene text localization and recognition. *PAMI*, 38(9):1872–1885, 2016. 2, 8

[18] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015. 3, 6

[19] S. Ren, K. He, R. B. Girshick, and J. Sun. Faster R-CNN: towards real-time object detection with region proposal networks. In *NIPS*, 2015. 1, 3

[20] A. Shrivastava, A. Gupta, and R. B. Girshick. Training region-based object detectors with online hard example mining. In *CVPR*, 2016. 6

[21] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. 2, 3

[22] Z. Tian, W. Huang, T. He, P. He, and Y. Qiao. Detecting text in natural image with connectionist text proposal network. In *ECCV*, 2016. 1, 2, 3, 7, 8

[23] K. Wang and S. J. Belongie. Word spotting in the wild. In *ECCV*, 2010. 2

[24] T. Wang, D. J. Wu, A. Coates, and A. Y. Ng. End-to-end text recognition with convolutional neural networks. In *ICPR*, 2012. 2

[25] C. Yao, X. Bai, W. Liu, Y. Ma, and Z. Tu. Detecting texts of arbitrary orientations in natural images. In *CVPR*, 2012. 1, 2, 6, 7

[26] C. Yao, X. Bai, N. Sang, X. Zhou, S. Zhou, and Z. Cao. Scene text detection via holistic, multi-channel prediction. *CoRR*, abs/1606.09002, 2016. 2, 3, 7

[27] X. Yin, W. Pei, J. Zhang, and H. Hao. Multi-orientation scene text detection with adaptive clustering. *PAMI*, 37(9):1930–1937, 2015. 7

[28] X. Yin, X. Yin, K. Huang, and H. Hao. Robust text detection in natural scene images. *PAMI*, 36(5):970–983, 2014. 7

[29] Z. Zhang, W. Shen, C. Yao, and X. Bai. Symmetry-based text line detection in natural scenes. In *CVPR*, 2015. 2, 3, 8

[30] Z. Zhang, C. Zhang, W. Shen, C. Yao, W. Liu, and X. Bai. Multi-oriented text detection with fully convolutional networks. In *CVPR*, 2016. 1, 2, 3, 7, 8